INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA INFORMÁTICA




# Plataforma eLearning e Biblioteca Multimédia

**DAVID CRUZ TRIGUEIRA**
Julho de 2019

POLITÉCNICO
DO PORTO

Instituto Superior de Engenharia do Porto

# Plataforma eLearning e Biblioteca Multimédia

**David Cruz Trigueira**

**Dissertação para obtenção do Grau de Mestre em**

**Engenharia Informática, Área de Especialização em Engenharia de Software**

**Orientador: Alberto Sampaio**

**Supervisor: Paulo Matos**

Porto, julho 2019

# Declaração de confidencialidade

Nos termos do contrato de estágio curricular e do acordo de confidencialidade celebrado com a ALERT Life Sciences Computing, S.A. (“ALERT”), o presente relatório é confidencial e poderá conter referências a invenções, know-how, desenhos, programas de computador, segredos comerciais, produtos, fórmulas, métodos, planos, especificações, projetos, dados ou obras abrangidos por direitos de propriedade industrial e/ou intelectual da ALERT. Este relatório só poderá ser utilizado para efeitos de investigação e de ensino. Qualquer outro tipo de utilização está sujeito a autorização prévia e por escrito da ALERT.

# Resumo

Ao longo dos anos, a exigência contínua do mercado da saúde e a evolução tecnológica têm provocado a expansão e evolução do *software* dedicado a essa área, tornando-o mais rico em conteúdo e em funcionalidades e, consequentemente, aumentando a sua complexidade de uso. Com o aumento dessa complexidade e com o tempo reduzido dos profissionais para se familiarizarem com esses tipos de *software*, é cada vez mais fundamental que existam ferramentas de fácil acesso que os auxiliem na sua formação sobre as funcionalidades de um sistema.

Sendo uma empresa no mercado da saúde, a ALERT desenvolveu um produto de eLearning no âmbito da sua suite de produtos para a formação e obtenção de certificados, porém falta-lhe um componente de carácter colaborativo dedicado à visualização, gestão e partilha de conteúdos entre utilizadores. Desta forma, a ALERT pretende adicionar uma Biblioteca Multimédia como complemento de apoio aos seus processos de ensino e aprendizagem. Esta Biblioteca permitirá centralizar conteúdos de aprendizagem num só local para facilitar a sua consulta e partilha, preservação, organização e promover o aspeto colaborativo de entreajuda entre os profissionais através dessa plataforma.

Assim, este documento apresenta os processos seguidos para o desenvolvimento de uma Biblioteca Multimédia a adicionar à suite de produtos da ALERT, incluindo o problema a resolver, o estudo do estado da arte de sistemas de Bibliotecas Digitais e a avaliação de possíveis ferramentas a usar no seu desenvolvimento, bem como a análise, arquitetura e implementação da solução desenvolvida. Considerando os resultados da Experimentação e Avaliação, a plataforma desenvolvida apresenta um grau de usabilidade classificado como "Excelente" e os utilizadores ficaram muito satisfeitos com o seu uso, pelo que a Biblioteca se encontra já disponível para os colaboradores da ALERT.

**Palavras-chave**: Java, Spring, REST, motor de pesquisa, Biblioteca Multimédia, eLearning

# Abstract

Over the years, the continuous demand of the health market and the technological evolution have caused the expansion and evolution of the *software* dedicated to this area, making it richer in content and in functionalities and, consequently, increasing its complexity of use. With the increasing complexity and time constraints of healthcare professionals to become familiar with these types of *software*, it is increasingly critical that there are easy-to-access tools that aid them in their formation about the functionalities of a system.

As a company in the healthcare business, ALERT has developed an eLearning product within its suite of products for training and certification, but it lacks to have a collaborative component dedicated to the visualization, management and sharing of content between users. To fill this gap, ALERT intends to add a Multimedia Library as a complement to support its teaching and learning processes. This Library will allow the centralization of learning contents in one place to facilitate their consultation and sharing, preservation, organization and promote the collaborative aspect of mutual aid between professionals using this platform.

Thus, this thesis presents the processes followed for the development of a Multimedia Library to be added to the ALERT product suite, including the problem to be solved, the study of the state of the art of Digital Libraries systems and the evaluation of possible tools to be used in its development, as well as the analysis, architecture and implementation of the developed solution. Considering the results of the Experimentation and Evaluation analysis, the developed platform presents a usability degree classified as “Excellent” and the users were very satisfied with its use, which led to making the Library available to all ALERT collaborators.

**Keywords**: Java, Spring, REST, search engine, multimedia library, eLearning

# Agradecimentos

Agradeço ao Instituto Superior de Engenharia do Porto (ISEP) por tão bem me ter acolhido durante todo este percurso académico e ter contribuído para a minha formação, pessoal e profissional, desde o início da licenciatura até ao fim do mestrado.

Agradeço ao Professor Alberto Sampaio por ter aceite ser o meu orientador e por todo o tempo dispendido e ajuda prestada durante este ciclo.

Agradeço a todos os colaboradores da ALERT pela sua forma de acolhimento, simpatia e disponibilidade, em especial ao meu supervisor, Paulo Matos, que me acompanhou em todas as fases de desenvolvimento deste projeto.

Por fim, agradeço aos meus amigos e família que sempre me apoiaram e incentivaram durante a minha formação académica.

# Índice

# Lista de Figuras

# Lista de Tabelas

# Lista de Extratos de Código

# Acrónimos e Símbolos

## Lista de Acrónimos

| | |
|---|---|
| **AD** | *Active Directory* |
| **AHP** | *Analytic Hierarchy Process* |
| **API** | *Application Programming Interface* |
| **CNRI** | *Corporation for National Research Initiatives* |
| **DC** | *Dublin Core* |
| **DEI** | Departamento de Engenharia Informática |
| **FFE** | *Fuzzy Front End* |
| **FURPS** | *Functionality, Usability, Reliability, Performance, Supportability* |
| **HTML** | *Hypertext Markup Language* |
| **HTTP/HTTPS** | *Hypertext Transfer Protocol/Hypertext Transfer Protocol Secure* |
| **ISO** | *International Organization for Standardization* |
| **JSON** | *JavaScript Object Notation* |
| **JSP** | *JavaServer Pages* |
| **JWT** | *JSON Web Tokens* |
| **LDAP** | *Lightweight Directory Access Protocol* |
| **MEI** | Mestrado em Engenharia Informática |
| **NCD** | *New Concept Development* |
| **NPD** | *New Product Development* |
| **OAI** | *Open Archives Initiative Protocol for Metadata Harvesting* |
| **PDF** | *Portable Document Format* |
| **QFD** | *Quality Function Deployment* |

| | |
|---|---|
| **RAP** | *Repository Access Protocol* |
| **REST** | *Representational State Transfer* |
| **RUP** | *Rational Unified Process* |
| **SOA** | *Service-Oriented Architecture* |
| **SOAP** | *Simple Object Access Protocol* |
| **SSL** | *Secure Socket Layer* |
| **SUS** | *System Usability Scale* |
| **TMDEI** | Tese/Dissertação/Estágio |
| **UNESCO** | *United Nations Educational, Scientific and Cultural Organization* |
| **URL** | *Uniform Resource Locator* |
| **VA** | *Value Analysis* |
| **VE** | *Value Engineering* |
| **XML** | *Extensive Markup Language* |
| **XSLT** | *Extensible Stylesheet Language Transformations* |

## Lista de Símbolos

| | |
|---|---|
| **λ** | *Lambda* |
| **®** | Marca registada |

# 1 Introdução

Este capítulo apresenta uma descrição sobre a contextualização do projeto, o problema inerente e os objetivos propostos. Para além disso, é também enunciada a motivação que conduziu à seleção deste projeto e os procedimentos adotados para atingir os seus objetivos, bem como a estrutura do restante documento.

## 1.1 Contexto

O presente documento foi desenvolvido no âmbito da unidade curricular de Tese/Dissertação/Estágio (TMDEI) do Mestrado em Engenharia Informática (MEI) do Instituto Superior de Engenharia do Porto (ISEP) em contexto de estágio na empresa ALERT.

O mercado da saúde, onde se insere a empresa ALERT para a qual foi realizado este projeto, é um mercado que solicita um elevado nível de exigência por lidar diretamente com a saúde dos cidadãos. O produto ALERT® eLEARNING promove as competências dos profissionais que lidam com a saúde dos pacientes e assume um papel fundamental no modelo de negócio da empresa, nomeadamente no processo de transferência de conhecimento e certificação de parceiros nacionais e internacionais, servindo também de meio de transmissão de conhecimento para todos os colaboradores da empresa.

O projeto "Plataforma eLearning e Biblioteca Multimédia" proposto pela ALERT foca-se no estudo e construção de uma Biblioteca Multimédia colaborativa baseada nos princípios da *Web* 2.0 (secção 2.1). Esta Biblioteca tem como objetivo apoiar o ensino e a aprendizagem proporcionado pela plataforma de eLearning, assumindo um papel de partilha, preservação e organização de conteúdos complementando os recursos de aprendizagem que apoiam os utilizadores.

## 1.2 Problema

A tecnologia encontra-se em constante evolução e desenvolvimento e, como consequência, a área da saúde torna-se mais exigente, requisitando a criação de novos produtos de *software* e a alteração dos já existentes com a implementação de inovações tecnológicas. Com a velocidade desta evolução, acresce a dificuldade de acompanhar todas as alterações e inovações, pelo que a criação de programas de eLearning e de conteúdos de aprendizagem assume um papel cada vez mais importante na rápida transmissão de conhecimento e formação das pessoas envolvidas. Este é um problema com que a ALERT se tem vindo a confrontar.

O produto ALERT® eLEARNING disponibiliza cursos para formação e obtenção de certificados, dispondo de uma área onde é possível rever os conteúdos dos módulos dos cursos frequentados. No entanto, falta-lhe um componente de carácter colaborativo dedicado à visualização, partilha e gestão de conteúdos, diferente e independente dos módulos dos cursos, de forma a complementar esses e outros temas com novas informações e facilitar o acesso às já existentes. Para além disso, este produto não apresenta uma interface de consulta de conteúdos amigável e adequada para aceder a partir de dispositivos móveis, o que é fundamental nos dias que correm. Os conteúdos resumem-se a fontes de informação representadas em diferentes formatos multimédia, como vídeos e apresentações PDF.

Assim, verificando o aumento da importância do eLearning e dos conteúdos de informação na ALERT e com a falta de uma área apropriada e específica de carácter colaborativo para a gestão e visualização desses conteúdos, emergiu a necessidade de desenvolver uma plataforma de conteúdos multimédia. Esta plataforma deve ser dinâmica, funcional e proporcionar uma boa usabilidade e fácil acesso, permitindo a interação e colaboração entre utilizadores, bem como poupar custos através da centralização dos conteúdos num só local de forma a facilitar a partilha de conhecimento entre os colaboradores e profissionais.

## 1.3 Objetivos

O objetivo principal deste projeto consiste em desenvolver uma nova plataforma *web* multimédia com o intuito de fornecer operações de gestão de diferentes tipos de conteúdo de aprendizagem (como a adição e remoção de conteúdos), disponibilizar e partilhar esse conteúdo e permitir a colaboração entre utilizadores, adotando características da *Web* 2.0.

De forma a atingir esse objetivo foram definidas as seguintes etapas a alcançar durante o projeto:

- Conhecer o estado da arte de repositórios e sistemas de Bibliotecas Multimédia existentes e de especificações relevantes na área adotadas por esses sistemas, bem como o estudo do estado da arte em tecnologia pertinente para o desenvolvimento do projeto;

- Definir os requisitos e a arquitetura para a nova plataforma *web*;

- Desenvolver uma nova plataforma, como solução final para o problema definido, com cuidado particular quanto aos requisitos relacionados com a usabilidade e com a apresentação de informação;

## 1.4 Partes Interessadas

As partes interessadas representam as pessoas, grupo de pessoas ou instituições com compromisso ou interesse num produto específico, não tendo necessariamente de estar envolvidos ou incluídos no processo de tomada de decisão, mas influenciam ou são influenciados de alguma forma pelo produto desenvolvido.

Considerando os objetivos especificados para a solução final do problema, a principal parte interessada são os colaboradores da ALERT e os utilizadores registados no produto ALERT® eLEARNING (profissionais de saúde em geral), sendo que todos os colaboradores são registados nesta plataforma quando são integrados na organização. Estes elementos são considerados partes interessadas uma vez que verão disponibilizados conteúdos de aprendizagem já existentes e adicionais aos da plataforma de eLearning, podendo assim melhorar o seu conhecimento de forma mais fácil e acessível num local próprio para consulta de conteúdos.

Outra parte interessada é a própria empresa. Para além de ser a proprietária do produto, tem interesse em ter um local próprio e adequado para chegar a todos os seus colaboradores e utilizadores do eLearning com novos conteúdos de forma mais prática e rápida, beneficiando não só por valorizar o produto ALERT® eLEARNING com a adição de mais um produto associado à aprendizagem e com conteúdos extra, mas também com a redução de recursos e tempo gasto para a disseminação de novas informações e com a melhoria do desempenho dos seus colaboradores.

## 1.5 Motivação Pessoal

No que diz respeito à motivação pessoal que levou ao interesse neste projeto, destaca-se o facto da proposta colocada se inserir na área de maior interesse pessoal nesta fase do meu percurso académico e profissional, mais concretamente a área de desenvolvimento de plataformas *web*, e o perfil do candidato requerido ser de *Full Stack Developer.* A área de atuação da organização no mercado da saúde também foi um fator de motivação na escolha efetuada, uma vez que esta é uma área que permite contribuir diretamente para a saúde e o bem-estar do ser humano, dando um maior propósito e motivação para o trabalho desenvolvido. Para além disso, esta é uma grande área de negócio que depende da tecnologia e dos sistemas de informação, pelo que a experiência adquirida nesta área pode ser uma mais

valia para o futuro. As condições de trabalho oferecidas e a integração e disponibilização dos colaboradores tiveram também peso na tomada de decisão.

## 1.6 Procedimentos

Para o desenvolvimento deste projeto foram definidas as metodologias de trabalho e as tarefas principais a serem executadas, promovendo uma gestão mais organizada de forma a conduzir ao cumprimento dos requisitos propostos.

### 1.6.1 Metodologia de trabalho

Durante o desenvolvimento do projeto procurou-se seguir a metodologia do *Rational Unified Process* (RUP). O RUP é um processo de engenharia de *software* definido pela International Business Machines (IBM) que segue uma abordagem iterativa disciplinada para atribuir tarefas e responsabilidades dentro de uma organização de desenvolvimento, tendo como objetivo a produção de *software* de alta qualidade que atenda às necessidades dos seus utilizadores finais (International Business Machines, 2001). Assim, seguiu-se um desenvolvimento iterativo das diferentes fases do projeto: domínio do problema, investigação, requisitos, análise de valor de negócio, análise e desenho da arquitetura representativa da solução para o problema, implementação da solução e realização de testes.

Aliado ao RUP foi também utilizado o Jira. O Jira é um *software* desenvolvido pela empresa Atlassian que permite a gestão, monitorização e acompanhamento de tarefas e atividades de um projeto (Atlassian, 2019). Com o uso desta plataforma é possível monitorizar e documentar registos associados a cada tarefa e ao estado do projeto, mantendo a organização e o supervisor atualizados.

### 1.6.2 Sistema de controlo de versões

O sistema de controlo de versões usado pela ALERT no desenvolvimento deste projeto para salvaguardar e manter o histórico das versões de desenvolvimento é o Apache Subversion (SVN). O Subversion é um sistema de controlo de versões de código aberto da Apache Software Foundation, caracterizado pela sua confiabilidade, simplicidade e segurança (Apache Software Foundation, 2018).

A aplicação cliente usada para trabalhar com o sistema SVN foi o TortoiseSVN, gratuito, intuitivo e fácil de usar evitando a necessidade de trabalhar com o SVN através da linha de comandos (TortoiseSVN, 2018).

### 1.6.3 Planeamento

De forma a organizar o fluxo de execução deste projeto foram estabelecidas as seguintes tarefas principais a desempenhar:

- Ganhar familiaridade com o mercado e conceitos acerca do tema eLearning, conteúdos de aprendizagem e Bibliotecas Multimédia;
- Analisar, interpretar e compreender o problema de forma a entender o contexto e as necessidades;
- Conhecer o estado da arte de repositórios e sistemas de Bibliotecas Multimédia existentes e de especificações relevantes na área adotadas por essas bibliotecas, quer a nível de arquitetura quer a nível de acessibilidade e usabilidade dos seus *websites*;
- Conhecer o estado da arte em ferramentas, metodologias e tecnologia relevantes nos sistemas em estudo;
- Realizar a análise de valor do produto a desenvolver;
- Elaborar a análise de requisitos funcionais e não funcionais;
- Definir a arquitetura da solução a implementar;
- Implementar a solução e respetivos testes ao *software*;
- Elaborar a experimentação e avaliação da usabilidade da plataforma desenvolvida.

## 1.7 Contribuições e Valor

Com o desenvolvimento deste projeto a ALERT irá alargar as potencialidades do eLearning, fortalecendo e valorizando o setor de aprendizagem e conhecimento da empresa ao complementar os conteúdos já existentes no produto ALERT® eLEARNING e disponibilizando novos. Através da nova plataforma desenvolvida é disponibilizado um repositório de itens de conteúdo de aprendizagem partilháveis e úteis, acedido a partir de qualquer local e dispositivo, necessitando apenas de acesso à internet e a um navegador.

Assim, este projeto contribui com os seguintes aspetos:

- Valorização do produto ALERT® eLEARNING com um produto associado à aprendizagem e com conteúdos extra;
- Centralização de fontes de informação num só local de acesso, o *website* da Biblioteca Multimédia;

- Disseminação de novas práticas, notícias importantes, tecnologias e procedimentos de forma rápida e eficaz, mantendo todos os colaboradores e utilizadores do eLearning atualizados com um simples acesso à nova plataforma. Este pode ser um meio de reduzir os custos envolvidos para a ALERT nessa atualização, evitando a possível necessidade de reuniões e/ou formações e outras formas de comunicação mais dispendiosas;

- Os colaboradores e utilizadores do eLearning da ALERT podem aprofundar e melhorar o seu desempenho consultando a plataforma quando quiserem, reduzindo os recursos gastos pela ALERT na necessidade de formações extra e melhorando a capacidade dos seus colaboradores;

- Assistência na procura de conteúdos, reduzindo o tempo de pesquisa comparativamente com a habitual navegação no sistema de ficheiros para encontrar o artefacto pretendido.

## 1.8 Estrutura do documento

Esta secção tem como objetivo apresentar a estrutura deste documento. Assim, o restante documento é composto pelos seguintes capítulos:

- Estado da Arte: este capítulo apresenta o conhecimento adquirido sobre sistemas de *software* de Bibliotecas Digitais existentes, nomeadamente as suas características a nível arquitetural e de especificações da área adotadas, seguido de uma comparação de algumas características destes sistemas. Para além disso, é apresentado também a análise e comparação de ferramentas tecnológicas relevantes na área em estudo, bem como uma exploração realizada a alguns *websites* de Bibliotecas Digitais para observar como estes apresentam visualmente as suas funcionalidades ao utilizador. No fim de cada estudo são indicadas algumas conclusões pertinentes para o produto desenvolvido;

- Análise de Valor: este capítulo apresenta a análise de valor do produto desenvolvido, que consiste na elaboração de um processo composto por diferentes técnicas para identificar as funções principais que proporcionam valor para o cliente, incluindo a utilização de métodos para auxílio no processo de tomada de decisão sobre possíveis tecnologias a utilizar no produto;

- Análise: neste capítulo são apresentados os requisitos funcionais e não funcionais a ter em conta na implementação da solução, o modelo de domínio e o modelo de dados do sistema, bem como maquetes da interface gráfica a adotar na plataforma;

- Arquitetura: aqui é exposta a arquitetura do sistema concebido através do seu diagrama de componentes e de implantação;

- Desenho e Implementação: este capítulo descreve o desenho e a implementação das funcionalidades do sistema tendo em conta a análise e arquitetura previamente definidas para a construção da solução, apresentando evidências da implementação efetuada com as respetivas explicações;

- Testes: neste capítulo é abordado como foram aplicados testes à solução implementada com exemplos dos mesmos;

- Experimentação e Avaliação: este capítulo apresenta a metodologia aplicada para experimentar e avaliar a usabilidade do produto final, expondo o tratamento efetuado aos dados recolhidos para obter o grau de usabilidade da plataforma e consequentemente elaborar conclusões;

- Conclusão: neste capítulo é apresentada uma observação sobre os objetivos concluídos, as limitações encontradas e o trabalho futuro a realizar, concluindo com uma apreciação final sobre o trabalho desenvolvido;

- Referências: neste capítulo são apresentadas as referências consultadas na elaboração deste documento;

- Anexos: no final deste documento são expostos os seus anexos.

# 2 Estado da Arte

Neste capítulo é apresentado o estudo do estado da arte efetuado sobre sistemas existentes relacionados com o tema do projeto e as tendências nele adotados, bem como um estudo e análise das potenciais tecnologias a usar.

## 2.1 Contextualização da Biblioteca Digital no conhecimento e aprendizagem

A tecnologia modificou e continua a modificar a vida das pessoas, incluindo a área da educação e distribuição de informações, revolucionando o processo de educação com novas formas de ensino e aprendizagem (Leonard, 2013). As Bibliotecas Digitais são ferramentas eficazes de disseminação de informação e conhecimento (Leonard, 2013) e, com o forte crescimento do eLearning em geral, cada vez mais é dada importância às Bibliotecas pois têm o potencial de oferecer uma quantidade enorme de recursos para apoiar o eLearning, tornando-se um componente essencial na qualidade que estes sistemas oferecem. Apesar desta importância, normalmente as Bibliotecas Digitais não estão diretamente integradas nos sistemas de gestão de aprendizagem, uma vez que são projetadas principalmente como repositórios de informação (Leonard, 2013). Assim, as Bibliotecas Multimédia normalmente são expostas como um sistema externo aos sistemas de eLearning, contribuindo e apresentando um grande potencial para disponibilizar recursos eletrónicos ilimitados, fornecendo serviços de informação que atendem às necessidades das partes interessadas.

Em 2004 a empresa americana O'Reilly Media popularizou o termo "Web 2.0" para se referir a uma segunda geração de comunidades e serviços oferecidos na internet. A este termo estão associados conceitos, tecnologias e tendências que visam aumentar a criatividade, comunicação, partilha, colaboração e a funcionalidade de plataformas *web,* enriquecendo a experiência do utilizador (Sastry & Reddy, 2010). Com o aparecimento da *Web* 2.0 e a

consequente transformação da internet para um canal de distribuição bidirecional e o meio digital como algo natural, a fronteira entre os produtores e consumidores de conteúdo tem vindo a desaparecer, possibilitando que os utilizadores tenham acesso à sua própria gestão e organização de informações, participando e influenciando ativamente o processo de aprendizagem e tornando-se eles próprios também produtores e publicadores de conteúdo (Al Owsi, 2010). Com a diminuição desta fronteira o conteúdo existente na internet aumentou significativamente, uma vez que é constituída uma oportunidade de produção de diversos recursos, mas também surge o desafio de gestão relacionado com a qualidade dos conteúdos dado que estes podem ser adicionados por utilizadores não profissionais na área de conteúdos de aprendizagem (Al Owsi, 2010). Assim, a *Web* 2.0 apresenta princípios importantes a adotar no desenho de Bibliotecas Digitais, assumindo um grande impacto na educação por potenciar e facilitar a obtenção de conhecimento.

Em suma, as Bibliotecas Digitais apoiam o ensino e a aprendizagem não só por desempenharem um papel na organização, preservação e partilha de conteúdo, mas também por reunirem pessoas com diferentes perspetivas e ideias (Fuchs, Muscogiuri, Hemmje, & Niederée, 2004).

## 2.2 Sistemas de *software* para Bibliotecas Digitais

Nesta secção é apresentada a análise efetuada a alguns sistemas de *software* usados para a construção de Bibliotecas Digitais. Primeiramente, são analisados individualmente alguns sistemas de forma a verificar como se encontram estruturados para servirem de repositórios de informação e, depois, é realizada uma comparação global das suas características. Nesta secção são também ainda expostas outras abordagens de arquiteturas de sistemas existentes e, por fim, é feita uma conclusão com base em todos os artefactos estudados. A partir da formulação desta conclusão, são referenciados potenciais fatores a ter em conta para a construção da nova plataforma, enunciando os aspetos comuns que se verificaram na análise destes sistemas.

Um *software* de Bibliotecas Digitais pode ser entendido como um *software* especializado para a construção de Bibliotecas Digitais, disponibilizando uma plataforma com funcionalidades para a sistemática seleção, aquisição, deposição, processamento, pesquisa e descoberta de documentos digitais (Rupesh, 2017). Os sistemas a seguir apresentados foram selecionados para análise com base no facto de serem os sistemas livres de código aberto mais usados no que diz respeito ao software de Bibliotecas Digitais (Pyrounakis & Nikolaidou, 2009) (Kökörčený, 2011) (Verma & Kumar, 2018).

### 2.2.1 Fedora

A arquitetura *Flexible Extensible Digital Object Repository Architecture* (Fedora) nasceu em 1997 na Universidade de Cornell e é uma arquitetura desenhada para armazenar, gerir e aceder a conteúdo digital. Esta arquitetura usa o conceito de objetos digitais inspirado nos princípios

da *framework* Kahn-Wilensky, uma *framework* para a distribuição de serviços de objetos digitais onde são definidos os elementos e componentes de uma Biblioteca Digital (Kökörčený, 2011). Na *framework* Kahn-Wilensky, relativamente à sua arquitetura, a interface do utilizador é separada dos repositórios, e cada repositório disponibiliza operações para aceder aos objetos digitais através do *Repository Access Protocol* (RAP).

Mais tarde, a Universidade da Virgínia uniu esforços com a Universidade de Cornell e foi desenvolvido o Fedora Repository, uma reimplementação de código aberto da arquitetura Fedora inicialmente desenhada na Universidade de Cornell (Staples, Wayland, & Payette, 2003). Desde 2009 este projeto é apoiado e desenvolvido pela organização sem fins lucrativos DuraSpace, que nasceu quando a organização responsável pelo Fedora se fundiu com a organização responsável pelo DSpace (DuraSpace, 2019).

A entidade básica do Fedora é o objeto digital. Um objeto digital é constituído por metadados e conteúdo digital, bem como ligações para objetos de comportamento associados a si (Pyrounakis & Nikolaidou, 2009). Um repositório Fedora disponibiliza o acesso aos objetos digitais através de ferramentas e serviços descritos nos objetos de comportamento. Os objetos de comportamento armazenam metadados que descrevem operações de uma ferramenta/serviço e as ligações para executar as operações (Staples, Wayland, & Payette, 2003). Na Figura 1 é demonstrado o modelo do *Fedora object model*.

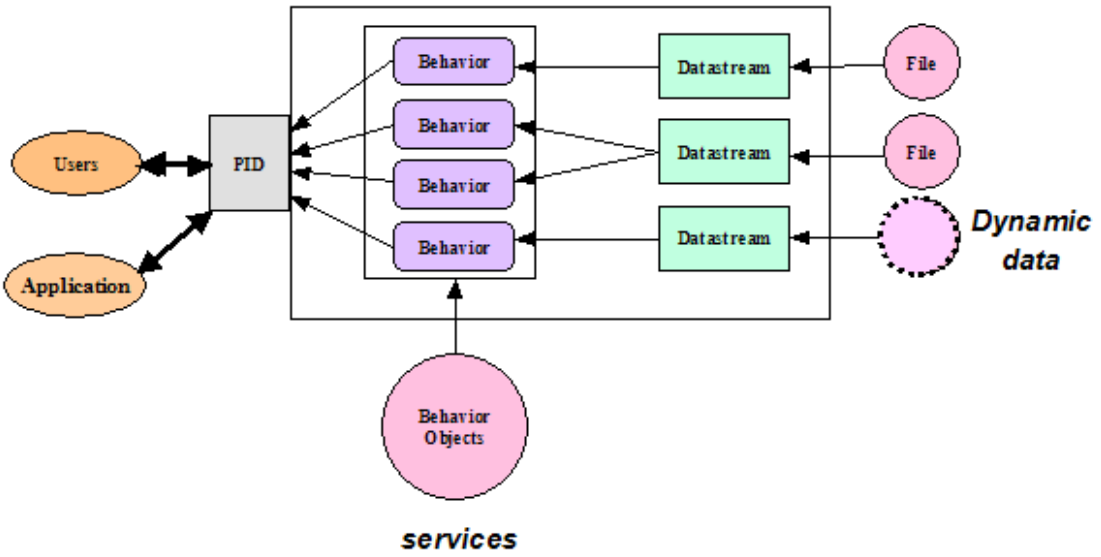



Figura 1 – *Fedora object model* (Staples, Wayland, & Payette, 2003)

Relativamente à criação de objetos digitais, este processo pode ser efetuado usando a interface cliente do administrador ou através de uma importação *batch* de ficheiros XML, isto é, uma importação sem interação com o utilizador efetuada através da recolha dos dados contidos nos ficheiros XML. A adição ou edição de metadados é disponibilizada através de um editor de texto no cliente de Administrador (Pyrounakis & Nikolaidou, 2009) e a sua estrutura segue o *standard* Dublin Core (DC).

Comparado com outros sistemas, o Fedora tem o ponto forte de ter suporte para armazenamento e acesso de múltiplas versões de um objeto digital no repositório, uma vez que preserva automaticamente todas as versões de um objeto (Pyrounakis & Nikolaidou, 2009).

Este sistema aplica os princípios SOA (*Service-oriented architecture*), expondo APIs (*Application Programming Interfaces)* como serviços *web* (Staples, Wayland, & Payette, 2003). Na Figura 2 encontra-se representado a arquitetura de um sistema Fedora.

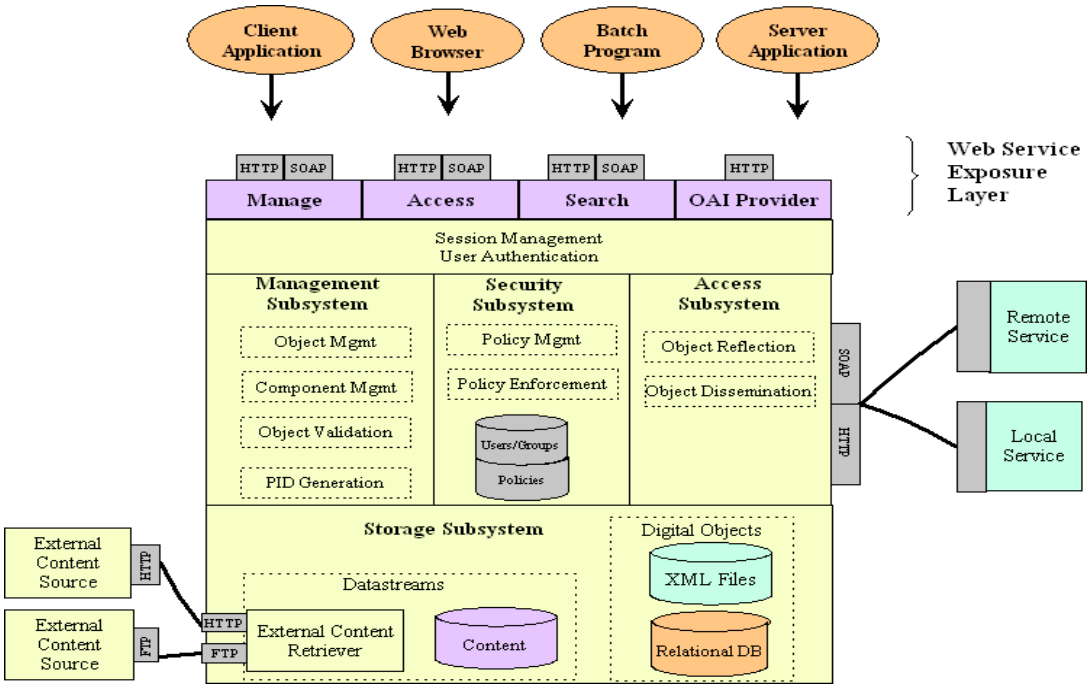



Figura 2 – Sistema Fedora Repository (Staples, Wayland, & Payette, 2003)

Um sistema Fedora é constituído por três camadas (Kökörčený, 2011):

- Camada de exposição dos serviços web;
- Camada principal do subsistema;
- Camada de armazenamento.

A primeira camada enunciada expõe quatro serviços *web* (The Fedora Development Team, 2008): o serviço de gestão, via HTTP (*Hypertext Transfer Protocol*) ou SOAP (*Simple Object Access Protocol*), é onde se encontram definidas as funcionalidades para a administração do repositório, incluindo as operações necessárias para os clientes criarem e gerirem objetos digitais; o serviço de acesso, disponibilizado também via HTTP ou SOAP, é onde se encontra definido o acesso aos objetos digitais armazenados no repositório; o serviço de pesquisa, exposto via HTTP ou SOAP e o serviço OAI (*Open Archives Initiative*), exposto via HTTP (Staples, Wayland, & Payette, 2003).

A camada principal do subsistema implementa a gestão e acesso aos subsistemas. O subsistema de gestão trata das operações de gestão dos objetos digitais (criação, edição, remoção, importação, exportação e manutenção) e da validação da integridade desses objetos. O subsistema de acesso é responsável pela disseminação do conteúdo dos objetos digitais (Staples, Wayland, & Payette, 2003).

A camada de armazenamento contém todas as operações relacionadas com a persistência dos objetos no repositório (Staples, Wayland, & Payette, 2003).

O armazenamento dos metadados é efetuado numa base de dados relacional, existindo suporte para diferentes bases de dados, sendo as mais comuns a Oracle, PostgreSQL e MySQL. O conteúdo digital pode ser armazenado no sistema de ficheiros ou numa solução na nuvem. Relativamente à pesquisa de conteúdo, este pode ser facilmente indexado utilizando aplicações de pesquisa como o Solr (DSpace, 2018).

### 2.2.2 DSpace

O DSpace é uma aplicação web de código aberto para colecionar, gerir, indexar e distribuir conteúdo digital. Direcionado para instituições da área académica e de pesquisa, este sistema foi originalmente desenvolvido numa cooperação entre o MIT (Massachusetts Institute of Tecnology) e a HP (Hewlett Packard) (Tramboo, Humma, & Shafi, 2012). Foi lançado em 2002 e desde aí, como projeto de código aberto, tem sido evoluído por uma comunidade global de desenvolvedores, gestores de repositórios e outras partes interessadas em contribuir no projeto, sendo um projeto apoiado diretamente desde 2009, tal como o Fedora, pela organização DuraSpace (DuraSpace, 2018).

Na Figura 3 é apresentado um modelo de informação do sistema DSpace.

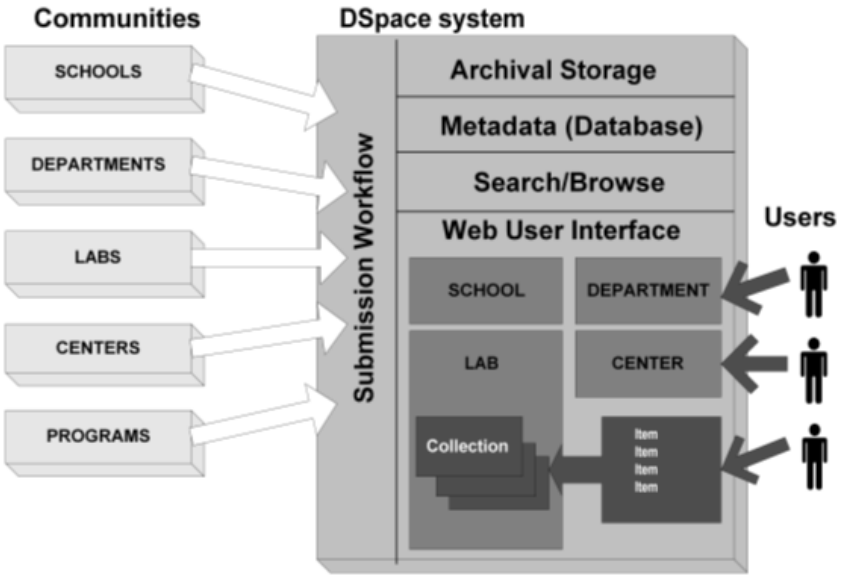



Figura 3 – Modelo de informação do DSpace (Smith, et al., 2003)

Como se pode visualizar, o modelo de informação do DSpace está construído sobre a ideia de comunidades e foi desenhado para que cada comunidade possa adaptar o sistema de acordo com as suas necessidades (Smith, et al., 2003). A comunidade é o nível mais alto na hierarquia do DSpace, pode corresponder a instituições ou partes de instituições como também apenas corresponder a áreas temáticas. Este sistema suporta coleções de itens e uma comunidade pode conter uma ou mais coleções. Um item pode pertencer a mais do que uma coleção, mas apenas pode ter uma coleção como "dono" (Pyrounakis & Nikolaidou, 2009). A entidade básica do sistema DSpace é designada de item. Um item consiste no conteúdo digital (em qualquer tipo de formato) e no agrupamento de informação relacionada com o ficheiro (conteúdo), isto é, metadados. Os campos de metadados seguem o standard Dublin Core qualificado (DuraSpace, 2018) que, comparativamente com o *standard* Dublin Core simples, possui a adição de alguns campos extra, como o *abstract* de documentos, *keywords*, entre outros. Os metadados são então armazenados nos itens e indexados para navegação e pesquisa do sistema. Cada item é identificado com uma identificação global única baseada no *Handle System* da CNRI (Corporation for National Research Initiatives) (Smith, et al., 2003).

A criação/submissão dos conteúdos é feita através da submissão de um utilizador na interface gráfica ou através de uma importação *batch*. Em ambos os casos é iniciado um *workflow* dependente da configuração existente. Um *workflow* pode ser configurado de forma a que os itens passem por um processo de aprovação de até três passos, percorridos por diferentes utilizadores ou grupos antes da efetivação da submissão. O *workflow* consiste nos seguintes passos (Kökörčený, 2011):

- Submissão;
- Aceitação ou rejeição;

- Rejeição ou aceitação com possível edição de metadados;
- Edição de metadados;
- Armazenamento

Este *workflow* pode ser visualizado na Figura 4.

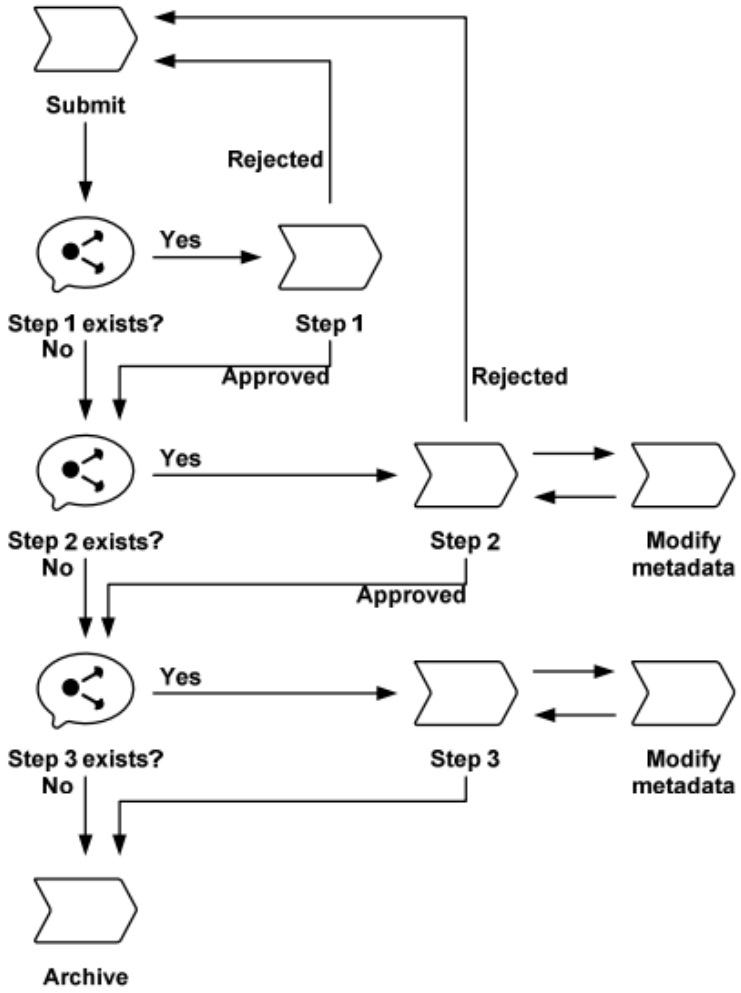



Figura 4 – Processo de aprovação de submissão no DSpace (Kökörčený, 2011)

Uma desvantagem ao adotar o sistema DSpace é o facto de este *workflow* de aprovação ser fixo e não ser possível modificá-lo nem definir o próprio *workflow* de aprovação (Kökörčený, 2011).

Entrando em detalhe técnico na arquitetura do DSpace, na Figura 5 é mostrada a estrutura do sistema.

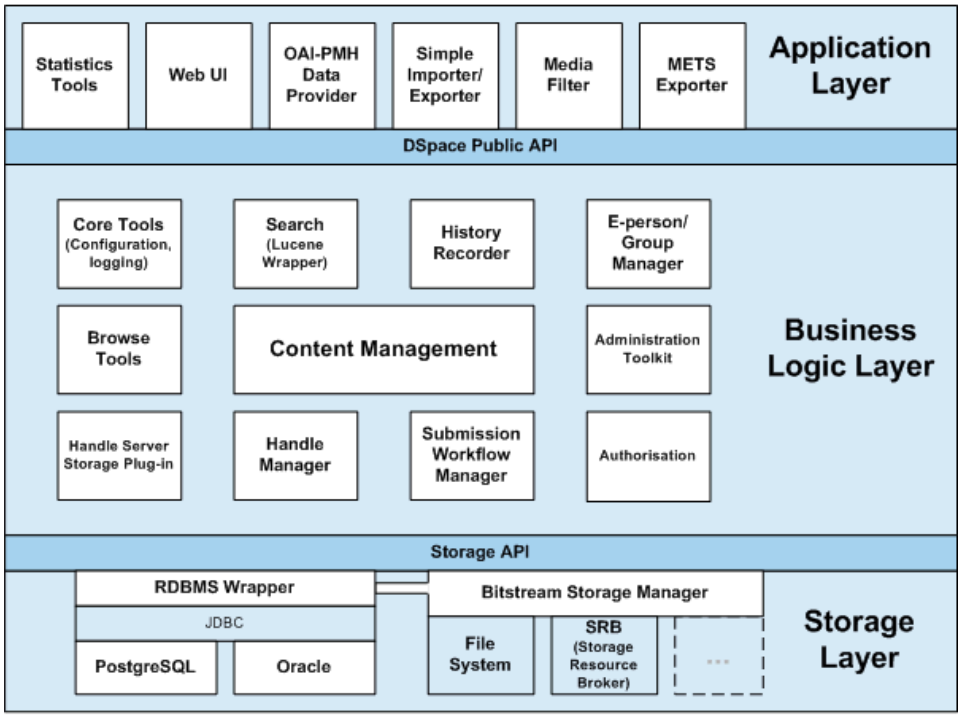



Figura 5 – Arquitetura do DSpace (Donohue, 2015b)

O DSpace é uma aplicação *full stack web* (DuraSpace, 2018), composta por três camadas: camada de armazenamento, camada de lógica de negócio e camada de aplicação. Cada camada apenas comunica com a camada abaixo de si. (Donohue, 2015b). Cada componente presente na camada de armazenamento e de lógica de negócio tem uma API definida e a união das APIs desses componentes é chamada *Storage API* (para a camada de armazenamento) e *DSpace Public API* (para a camada de lógica de negócio).

A camada de armazenamento é responsável pelo armazenamento dos conteúdos digitais e dos metadados. O módulo de armazenamento denominado *Bitstream Storage Manager* armazena os conteúdos digitais por predefinição no sistema de ficheiros, enquanto os metadados e outros dados são guardados numa base de dados relacional Oracle ou PostgreSQL (DuraSpace, 2018).

A camada de lógica de negócio é constituída pelos módulos que representam as principais funcionalidades do sistema, como a gestão de conteúdo, o *workflow* de submissão, ações de administração, a navegação e pesquisa de conteúdos, entre outros. A API de gestão de conteúdo é a API mais usada pelos componentes presentes na camada de aplicação e contém classes Java para ler e manipular conteúdo armazenado no sistema (Donohue, 2016c). Relativamente à API de pesquisa, inicialmente este sistema indexava campos de metadados recorrendo ao motor de pesquisa Lucene (Pyrounakis & Nikolaidou, 2009) mas, atualmente, vem predefinida com o motor de pesquisa Solr, baseado no Lucene, com funções de pesquisa e navegação mais avançadas (DuraSpace, 2018).

A camada de aplicação contém os componentes responsáveis por estabelecer a comunicação com o mundo exterior ao sistema. O componente de interface de utilizador (*Web* UI) é o

componente mais usado nesta camada e permite que os utilizadores acedam ao DSpace através dos seus navegadores de internet (Donohue, 2015a). O DSpace suporta dois tipos de interfaces de utilizador, JSP (*JavaServer Pages*) e XMLUI (*Apache Cocoon framework)* (Donohue & Pottinger, 2017). Outros dos componentes desta camada são o OAI-PMH (*Open Archives Iniative protocol for Metada Harvesting*) *Data provider* para partilha de metadados (Donohue, 2015b), e o *Simple Importer/Exporter* que possui ferramentas da linha de comandos para importar e exportar conjuntos de itens (Donohue, 2017d).

Relativamente aos aspetos menos positivos deste sistema tendo em conta as configurações predefinidas com que é disponibilizado, verificam-se algumas limitações ao nível de escalabilidade e extensibilidade do sistema (Verma & Kumar, 2018), a sua interface gráfica é fixa e pouco apelativa (Verma & Kumar, 2018) e são permitidas poucas intervenções para modificá-la (Tramboo, Humma, & Shafi, 2012) (Pyrounakis & Nikolaidou, 2009), sendo que a interface apresentada para a secção de administração é pouco sofisticada e intuitiva (Donohue, 2015a). O processo de aprovação de conteúdos submetidos é inflexível e não é permitida a construção de objetos muito diferentes devido à estrutura dos metadados ser limitada, uma vez que a sua arquitetura é orientada a base de dados (Tramboo, Humma, & Shafi, 2012).

### 2.2.3 EPrints

O EPrints é um *software* de código aberto desenvolvido pela Universidade de Southampton para construir repositórios de fontes digitais (Kökörčený, 2011). O repositório Eprints é um sistema de gestão de conteúdo configurável, configurado com maior ênfase para repositórios institucionais e jornais científicos, mas pode ser configurado e usado facilmente com outros tipos de conteúdos digitais, como imagens e áudio (Tramboo, Humma, & Shafi, 2012). É, portanto, um sistema de gestão de conteúdo configurável e extensível.

A entidade básica do EPrints é um objeto de dados que consiste num registo com metadados. Cada objeto de dados tem um identificador único e, ao contrário de outros sistemas, um ou mais ficheiros (documentos) podem estar ligados a si. No Eprints os objetos de dados são agrupados através de campos específicos, como a data e o título, não existindo o conceito de coleções e relações entre documentos (Pyrounakis & Nikolaidou, 2009).

Este sistema caracteriza-se por ser de fácil utilização para o utilizador, sendo a criação de objetos de dados efetuada através da interface gráfica. Os utilizadores simplesmente inserem os metadados como o título, tipo, autor, nome, entre outros. A estrutura dos metadados pedidos pode ser configurada pelo administrador e os utilizadores, caso tenham permissões, podem editar e remover documentos após a sua submissão (Pyrounakis & Nikolaidou, 2009).

O EPrints está configurado para ter três papéis de utilizadores diferentes (Tramboo, Humma, & Shafi, 2012):

- Administrador – controla a aparência da interface gráfica e todas as restantes operações e componentes do lado do servidor;

- Editor - revê as submissões antes de serem publicadas online e pode editar metadados de submissões;

- Autor - pode submeter documentos e gerir documentos previamente submetidos por si.

Relativamente ao armazenamento dos dados, os objetos de dados, que contém os metadados dos documentos, são armazenados numa base de dados MySQL, enquanto os documentos (conteúdo digital) são armazenados no sistema de ficheiros (Pyrounakis & Nikolaidou, 2009).

Para efeitos de pesquisa, de forma a aumentar o desempenho do tempo de procura, os campos de metadados desejados podem ser indexados na base de dados MySQL. De forma a encontrar a informação desejada mais rapidamente, o utilizador tem a opção combinada de pesquisa por seleção de campos e de texto livre, e a navegação é feita com base na escolha de filtros que correspondem a campos de metadados (Pyrounakis & Nikolaidou, 2009).

Quanto às suas limitações, o EPrints não tem nenhuma função que permita efetuar o envio de grandes conjuntos de arquivos ao mesmo tempo, permitindo apenas efetuar a submissão de vários arquivos quando estes pertencem ao mesmo registo (Verma & Kumar, 2018). Ao nível da sua funcionalidade de pesquisa o EPrints deixa a desejar, uma vez que a pesquisa baseada em pesquisa booleana não está disponível e por vezes não são apresentados resultados de uma dada pesquisa nem sugestões de pesquisa alternativas (Verma & Kumar, 2018).

### 2.2.4 Greenstone

O Greenstone Digital Library (GSDL) é um *software* de código aberto desenvolvido pela cooperação entre o New Zealand Digital Library Project na Universidade de Waikato, a UNESCO (Organização das Nações Unidas para a Educação, a Ciência e a Cultura) e a The Human Info NGO (Greenstone, 2015b). O objetivo deste *software* é capacitar os utilizadores para construírem as suas próprias Bibliotecas Digitais, disponibilizando para isso formas de organização e publicação de informação na internet ou em média removível (como *CDs* e *USB drives*) na forma de recursos digitais representados por metadados e totalmente pesquisáveis (Greenstone, 2015a). Existem duas versões principais deste *software*, o Greenstone 2 e o Greenstone 3. O último (Greenstone 3) está escrito em linguagem Java e usa tecnologias *web,* consistindo numa reimplementação e redesenho do *software* originalmente desenvolvido em 2000, o Greenstone 2 (Greenstone, 2015a). O Greenstone 3 encontra-se atualmente em desenvolvimento e, de forma a facilitar a transição do utilizador entre versões, é compatível com a versão anterior e inclui todas as suas funcionalidades (Greenstone, 2015a).

No Greenstone 3 os dados são divididos em documentos e em recursos. Os documentos representam os dados primários expressos em formato XML em que se baseia a indexação e

navegação, enquanto os recursos representam o conteúdo digital como ficheiros de vídeo, áudio e PDF. Cada recurso fica associado a um documento e, uma vez que só os documentos são indexados, os recursos só podem ser descobertos através da sua ligação a um documento (Don, Bainbridge, & Witten, 2002). O uso de XML combinado com transformações XSL (XSLT) para a representação de dados contribui para um mecanismo flexível, possibilitando modificações a funcionalidades do sistema em tempo de execução sem ser necessário recompilar o seu código fonte (Don, Bainbridge, & Witten, 2002).

O Greenstone é constituído por coleções que por sua vez são constituídas por documentos de informação. Uma coleção pode conter documentos textuais, imagens, áudio e vídeo, sendo que os materiais não textuais ficam associados a documentos ou descrições textuais. Para cada coleção criada é possível definir um conjunto de características de configuração (Tramboo, Humma, & Shafi, 2012) que descrevem a sua funcionalidade, como características de indexação, opções de pesquisa e navegação, formatos de ficheiros, entre outros (Pyrounakis & Nikolaidou, 2009). A indexação é efetuada a documentos de texto e campos específicos de metadados, já a pesquisa pode ser feita por palavras presentes em todo o documento de texto indexado e por secções definidas no documento, como o título, capitulo e parágrafo (Pyrounakis & Nikolaidou, 2009). Para além destas opções é também possível procurar por certos campos de metadados associados aos documentos (Tramboo, Humma, & Shafi, 2012). As opções de navegação podem ser definidas para campos específicos como o título ou assunto (Pyrounakis & Nikolaidou, 2009).

Relativamente aos formatos de metadados suportados, o Greenstone disponibiliza por predefinição vários conjuntos de formatos de metadados, apresentando-os numa interface gráfica para que os utilizadores possam escolher o formato desejado. Esses conjuntos existentes por definição são o Dublin Core simples e qualificado, o RFC 1807, o NZGLS (New Zealand Government Locator Service) e o AGLS (Australian Government Locator Service). Este sistema permite também que os utilizadores definam novos formatos de metadados através da disponibilização de uma opção de edição de metadados. Para além destas opções, o Greenstone utiliza *plugins* para suportar metadados definidos externamente de diferentes formas, como METS, DSpace, CSV, XML, OAI, entre outros (Greenstone, 2015a). O armazenamento dos ficheiros XML que contêm os metadados (documentos) são armazenados no sistema de ficheiros, assim como os recursos relativos aos conteúdos digitais (Pyrounakis & Nikolaidou, 2009).

No que toca à interface de utilizador, o Greenstone tem duas interfaces interativas diferentes, sendo que uma delas é destinada aos leitores e outra aos bibliotecários. A primeira interface gráfica destina-se aos utilizadores finais e é acedida através do navegador de internet. A segunda interface permite efetuar as operações relativas à gestão de coleções e configurações, como a adição de metadados, a criação e/ou seleção das opções de pesquisa e navegação que a coleção irá disponibilizar ao utilizador, entre outras (Greenstone, 2015a). As novas coleções e os seus documentos são construídos através do uso desta interface (Pyrounakis & Nikolaidou, 2009).

Na Figura 6 é apresentado um esquema de como funciona o sistema Greenstone.

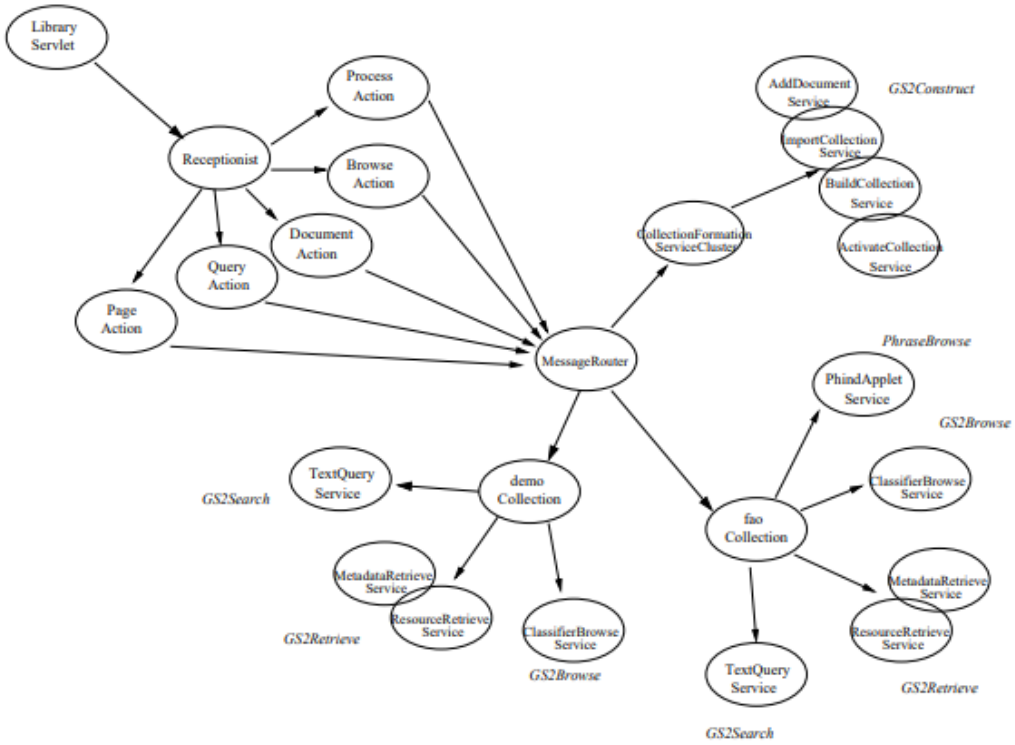



Figura 6 – Funcionamento do sistema Greenstone (Don, Bainbridge, & Witten, 2002).

O Greenstone 3 é constituído por um *back-end* que contém a lógica de negócio do sistema com os serviços necessários para efetuar a gestão de coleções e documentos e por um *front-end,* responsável por disponibilizar a interface gráfica ao utilizador através da internet e pela apresentação, pesquisa e navegação das coleções e documentos (Pyrounakis & Nikolaidou, 2009). Na Figura 6 o *front-end* é chamado de *receptionist* e este elemento comunica com o *back-end* através do agente *MessageRouter* (MR). O *MessageRouter* é um módulo do sistema por onde passam todas as comunicações efetuadas com o *back-end* (Don, Bainbridge, & Witten, 2002). Neste *software*, cada coleção é organizada individualmente e contém um aglomerado de serviços (*service cluster*) que agrupa um conjunto de serviços relacionados com o conjunto de dados fornecidos. A comunicação inicial com o utilizador é efetuada através do protocolo HTTP sobre a forma de URLs e argumentos. Estes comandos HTTP são recebidos e traduzidos para formato XML através do *Library Servlet* e interpretados pelo *receptionist*. A partir daí o *receptionist* executa diferentes ações efetuando chamadas ao *MessageRouter* (Don, Bainbridge, & Witten, 2002). Um ponto negativo da representação da interface do sistema em formato XML é eficiência de execução, uma vez que a informação transmitida requer transformações para ser interpretada (Don, Bainbridge, & Witten, 2002). O exemplo demonstrado na Figura 6 representa um exemplo da estrutura do Greenstone em que existe apenas um *back-end* envolvido, mas também é possível efetuar a troca de informações com sistemas a executarem em máquinas diferentes. Neste caso, os *MessageRouters* têm a

capacidade de comunicar numa rede distribuída com outros *MessageRouters* através de um protocolo definido (Don, Bainbridge, & Witten, 2002) (Tramboo, Humma, & Shafi, 2012).

### 2.2.5 Comparação

Na Tabela 1 é apresentada uma comparação de algumas características dos três sistemas anteriores com base na revisão da literatura efetuada e na documentação oficial destes (Greenstone, 2018) (Mark & Woods, 2018) (Becker, 2018).

Tabela 1 - Características dos sistemas de *software* de Bibliotecas Digitais. Adaptado de Tramboo, Humma e Shafi (2012), Verma e Kumar (2018), e Rupesh (2017).

| | Fedora | DSpace | EPrints | Greenstone |
|---|---|---|---|---|
| Ano de criação | 1998 | 2002 | 2000 | 1997 |
| Licença | Apache 2.0 | BSD | BSD | GNU |
| Formatos de metadados predefinidos | Dublin Core, Dublin Core qualificado, METS | Dublin Core, Dublin Core qualificado | Dublin Core, METS | Dublin Core, Dublin Core qualificado, RFC 1807, NZGLS (New Zealand Government Locator Service), AGLS (Australian Government Locator Service) |
| Funções de pesquisa | Por campo específico, lógica booleana | Por campo específico, lógica booleana, opções de classificação/ordenação | Por campo específico, opções de classificação/ordenação | Por campo específico, lógica booleana. |
| Opções de navegação | - | Por autor, título, assunto e coleção | Navegação pode ser feita a partir de qualquer campo especificado | Navegação pode ser feita a partir de qualquer campo |

| | Fedora | DSpace | EPrints | Greenstone |
|---|---|---|---|---|
| | | | | especificado |
| OAI-PMH | Sim | Sim | Sim | Sim |
| Tipos de conteúdos suportados (Armazenamento e fornecimento) | Armazena qualquer tipo | Armazena qualquer tipo | Armazena qualquer tipo | Armazena qualquer tipo |
| Interface gráfica predefinida | - | Sim | Sim | Sim |
| Base de dados | MySQL, Oracle, PostgreSQL | Oracle, PostgreSQL | MySQL | Gnu Database Manager |
| Servidor web | Servidor Apache, Apache Tomcat, Jetty | Servidor Apache, Apache Tomcat | Servidor Apache | Servidor Apache, IIS |
| Autenticação de utilizadores | Integração com vários sistemas de autenticação como LDAP e Shibboleth | Sistema próprio de autenticação, possível integração com vários sistemas como LDAP e Shibboleth | LDAP | Grupos de utilizadores |
| Pré-visualização de imagem em miniatura (thumbnail) | - | Imagens | Imagens, áudio, vídeo | Imagens, áudio, vídeo |
| Extensão de *plugins* | Sim | Sim | Sim | Sim |

Os três sistemas analisados cumprem os requisitos gerais das Bibliotecas Digitais apesar de diferirem na sua arquitetura e apresentação. Como tal, a seleção de um destes sistemas como plataforma para uma Biblioteca Digital deve ser efetuada com base nas metas e especificações de cada Biblioteca em particular. Assim sendo, dependendo das necessidades, um destes sistemas pode ser preferido sobre outro, pelo que quem pretende adotar um deve avaliar as suas coleções e a sua estratégia de distribuição de pesquisa com o intuito de escolher a plataforma que melhor satisfaça os seus objetivos (Bankier & Gleason, 2014). Normalmente, a seleção do *software* é baseada no tipo/formato do conteúdo a ser enviado, na forma como este

será divulgado e distribuído, no *front-end* e *back-end* do *software* para efetuar possíveis alterações e no tempo disponível para configurações (Verma & Kumar, 2018).

### 2.2.6 Outras abordagens

Existem outros sistemas que diferem na conceção da arquitetura de *software* para Bibliotecas Digitais. A arquitetura seguinte é baseada em conceitos gerais de arquiteturas de Bibliotecas Digitais expostas por investigadores. Consiste numa visão geral de uma arquitetura adotada num projeto iniciado pelo parlamento português para construir uma Biblioteca Digital. O objetivo era que a Biblioteca permitisse armazenar os registos da assembleia sobre a forma de diferentes tipos de documentos multimédia (Zagalo, Martins, Almeida, & Pinto, 2003).

Em geral, a arquitetura seguida neste projeto do parlamento segue conceitos já expostos de arquiteturas aplicadas noutras Bibliotecas Digitais. A funcionalidade global do sistema é particionada em vários serviços e cada um dos serviços tem uma interface pública definida. Essas interfaces de cada serviço definem os pedidos que são permitidos e as possíveis respostas e exceções (Zagalo, Martins, Almeida, & Pinto, 2003).

Tendo isso em conta, o sistema projetado é baseado nos princípios SOA. Isto significa que a funcionalidade da Biblioteca Digital é representada pela união de todas as funcionalidades oferecidas pela agregação individual de cada serviço (Zagalo, Martins, Almeida, & Pinto, 2003). Um sistema assim desenhado é um sistema muito mais flexível que os sistemas monolíticos[1]. A arquitetura deste sistema pode ser segmentada em três camadas, comunicando entre si através dos serviços *web* (Zagalo, Martins, Almeida, & Pinto, 2003):

- Camada front-end;
- Camada de indexação e *query*;
- Camada de repositório.

Na Figura 7 é apresentada uma arquitetura geral de sistemas de Bibliotecas Digitais baseados em serviços.

[1] Uma aplicação de *software* de camada única que contém todas as funcionalidades necessárias para o seu funcionamento.

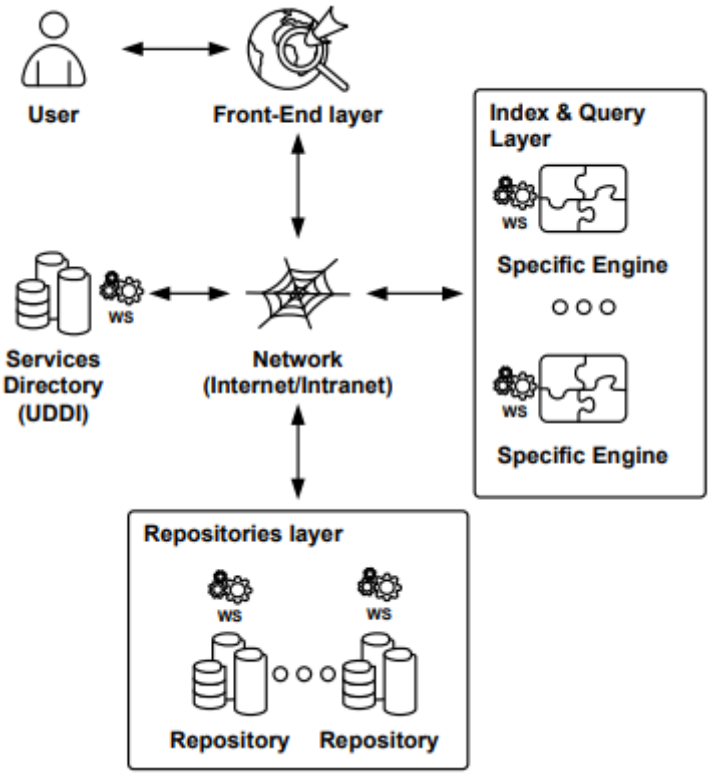



Figura 7 – Arquitetura geral de Bibliotecas Digitais baseada em serviços (Kökörčený, 2011)

A camada *front-end* divide-se em duas interfaces diferentes, em que a primeira representa a interface gráfica do utilizador, acessível através de um navegador, e a segunda representa a interface computacional, acessível através de serviços *web e* responsável por processar todos os pedidos do utilizador e do sistema (Kökörčený, 2011). Assim, uma das responsabilidades da primeira interface é a interação com todos os clientes do sistema, enquanto a outra interface é a componente responsável por distribuir os pedidos e recolher e fornecer os resultados obtidos em resposta a esses pedidos. Os clientes do sistema podem ser utilizadores a aceder ao sistema a partir do navegador ou agentes computacionais em comunicação com o sistema como um serviço (Zagalo, Martins, Almeida, & Pinto, 2003).

A camada de indexação e *query* tem como principal função disponibilizar funções de indexação e pesquisa sobre todas as informações armazenadas na camada de repositório, usando para isso motores específicos que suportam essas operações. Recorrendo a estes motores são obtidas as informações que são retornadas em resposta às *queries* submetidas pelo sistema. Esses resultados podem ser retornados sobre diferentes formas, dependendo da especificação existente na *query* submetida, como a forma de metadados ou apenas a identificação do documento que permita a sua recolha do repositório (Zagalo, Martins, Almeida, & Pinto, 2003).

A camada de repositório é constituída por múltiplos repositórios de informação que podem ser representados por bases de dados ou simplesmente por sistema de ficheiros. Esta camada disponibiliza funcionalidades para armazenamento e acesso a informação que, neste caso, são documentos multimédia (imagens, som, vídeo, etc.). Para a implementação dos repositórios,

apesar de muitas bases de dados relacionais disponibilizarem suporte para dados XML, foi selecionada para este projeto uma base de dados XML nativa (Xindice) (Zagalo, Martins, Almeida, & Pinto, 2003).

Na Figura 7 onde se encontra representada a arquitetura, encontra-se um componente chamado *Services Directory*. Este componente é um serviço que permite o registo de todos os serviços *web* existentes e o acesso às suas funcionalidades, atuando como uma espécie de orquestrador. É baseado no *standard Universal Description, Discovery and Integration* (UDDI) e todos os serviços registados são descritos através da *Web Services Description Language* (WSDL). Quando um serviço deseja comunicar com outro, deve aceder primeiramente ao *Services Directory* para saber com qual deve comunicar e como. Desta forma os serviços não têm conhecimento da existência dos outros, à exceção do serviço disponibilizado pelo *Services Directory* (Zagalo, Martins, Almeida, & Pinto, 2003). Assim, é implementado o padrão *Indirection* que promove o baixo acoplamento entre serviços, tornando o sistema mais flexível, dinâmico e personalizável. A comunicação entre serviços *web* é feita usando tecnologias *standard* da altura em que o projeto foi desenvolvido, nomeadamente o protocolo HTTP, SOAP e XML (Kökörčený, 2011).

### 2.2.7 Conclusões

Com base no estudo dos sistemas analisados, pode-se concluir que há alguns fatores que esses sistemas têm comum e que devem ser considerados na conceção da nova solução:

- Há uma separação definida de responsabilidades, com a arquitetura dos sistemas a ser composta por diferentes camadas e/ou orientada a serviços (SOA);
- Os sistemas de Bibliotecas Digitais usam o conceito de metadados para descrever a informação dos conteúdos multimédia armazenados, independentemente do tipo de formato do conteúdo (texto, imagem ou vídeo) e para melhorar as funcionalidades de pesquisa;
- Os metadados são, normalmente, armazenados num sistema de base de dados;
- O *standard* mais comum que serve de base para representar a estrutura dos metadados é o Dublin Core;
- Os conteúdos digitais são normalmente armazenados no sistema de ficheiros*;*
- Para melhorar o desempenho de indexação e pesquisa rápida de informação sobre os conteúdos, para além da indexação de campos de metadados na base de dados, grande parte destes tipos de sistemas recorrem ao uso de motores de pesquisa específicos para esse efeito, indexando os metadados relevantes de cada conteúdo;

- As opções de filtragem da pesquisa disponibilizadas para seleção do utilizador correspondem a campos de metadados indexados.

## 2.3 Ferramentas de pesquisa

Com o crescimento súbito da internet e os repositórios digitais, a rápida evolução das tecnologias, como as redes sociais, *cloud computing*, aplicações móveis e *big data*, um dos principais desafios na arquitetura de *software* é lidar com o grande volume de dados produzido e consumido pelos utilizadores (Grainger & Potter, 2014).

É esperado que qualquer aplicação que contenha dados para exposição disponibilize uma funcionalidade de pesquisa para encontrar o que se procura, sem ser necessário percorrer todo o *website* (Hinman, Gheorghe, & Russo, 2015). As aplicações atuais lidam com a geração e gestão de dados em grandes proporções diariamente, e tendo isso em conta, o desempenho exigido na utilização desses dados continua a aumentar, sendo que os utilizadores esperam cada vez mais um maior desempenho, disponibilidade e resposta do sistema (Grainger & Potter, 2014). À medida que os dados se vão acumulando, pesquisar sobre toda essa informação de forma eficiente e eficaz tornou-se um desafio.

Com a evolução tecnológica, tem surgido um interesse crescente em tecnologias de armazenamento e processamento de dados não relacionais especializadas, conhecidas como *NoSQL*. Estes sistemas partilham um padrão de desenho comum de combinar mecanismos de armazenamento e processamento com tipos específicos de dados, em vez de forçar todos os dados num modelo relacional comum (Grainger & Potter, 2014).

Nas secções seguintes são apresentados e comparados dois dos motores de pesquisa de código aberto mais famosos e utilizados no mundo, o Solr e o Elasticsearch (DB-Engines, 2018a). Ambos são construídos tendo como base uma biblioteca de pesquisa previamente existente, o Lucene, pelo que inicialmente é feita uma contextualização dessa biblioteca.

### 2.3.1 Apache Lucene

O Apache Lucene é uma poderosa biblioteca de pesquisa Java desenvolvida por Doug Cutting em 1997-1998 que permite adicionar funções de pesquisa a qualquer aplicação. É uma biblioteca de recolha de informações escalável e de alto desempenho. A recolha de informações refere-se ao processo de pesquisar por documentos, informações em documentos, ou metadados sobre documentos. É um projeto maduro e de código aberto, da Apache Software Foundation, licenciado sobre a licença de *software* Apache. Assim, o Lucene tem sido há já longos anos uma das bibliotecas mais populares de recolha de informações (Gospodnetic, Hatcher, & McCandless, 2010).

O Lucene é uma ferramenta que indexa elementos para realizar pesquisas sobre elementos indexados (Correia, 2016). Para pesquisar grandes quantidades de texto rapidamente, primeiro

é necessário indexar esse texto e convertê-lo num formato que permita efetuar uma pesquisa rápida. Este processo de conversão é chamado de indexação e o seu resultado de índice. Desta forma, pode-se pensar num índice como uma estrutura de dados que permite acesso rápido a palavras aleatoriamente guardadas dentro desse índice. O conceito aplicado pode ser comparado com um índice de um livro que permite localizar rapidamente as páginas que abordam determinados tópicos. O Lucene tem um sistema de indexação de índice invertido, o que significa que em vez de cada documento guardar os termos que nele existem, guarda, para cada termo, os documentos onde estes se encontram (Correia, 2016). Na Figura 8 são apresentados os componentes gerais de uma aplicação de pesquisa, sendo que os componentes a sombreado são as partes manipuladas pelo Lucene.

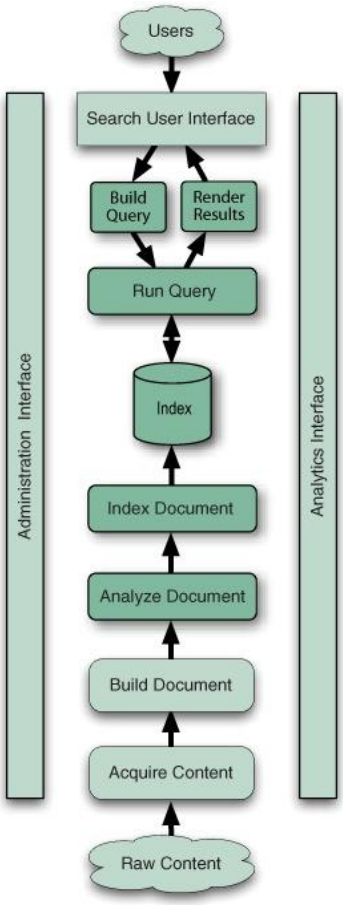



Figura 8 – Componentes típicos de uma ferramenta de pesquisa (Gospodnetic, Hatcher, & McCandless, 2010).

Na parte inferior da Figura 8 estão representados os componentes *Raw Content, Acquire Content* e *Build Document*. Estes componentes são relativos à obtenção e construção do documento a indexar no Lucene. Este documento normalmente pode ter origem numa base de dados, numa aplicação *web*, numa construção manual ou no sistema de ficheiros. De seguida, o Lucene recebe o documento, analisa-o e procede à sua indexação: conceito de processamento dos dados originais gerando referências cruzadas, altamente eficiente para facilitar a pesquisa rápida (Gospodnetic, Hatcher, & McCandless, 2010).

Na parte superior da Figura 8 encontram-se os componentes que acedem aos serviços de recolha de informação do Lucene. O utilizador, usando um navegador *web*, uma aplicação *desktop* ou um dispositivo móvel, interage com o Lucene inserindo os critérios de pesquisa que pretende. Esse pedido de pesquisa é submetido para o servidor, onde deve ocorrer a tradução desse pedido para um objeto de consulta reconhecido pelo Lucene. Esta tradução está presente no componente *Build Query* (Gospodnetic, Hatcher, & McCandless, 2010). Após esta tradução, a fase de *Run Query* encarrega-se de executar a *query*, pesquisando nos índices de pesquisa e retornando os documentos que correspondem aos critérios de pesquisa da *query*, ordenando-os pelo pedido de ordenação pretendido. Existem três modelos teóricos comuns de pesquisa (Gospodnetic, Hatcher, & McCandless, 2010):

- Modelo booleano puro - os documentos correspondem ou não à *query* fornecida;
- Modelo de espaço vetorial - ambas as *queries* e os documentos são modelados como vetores num espaço dimensional, onde cada termo exclusivo é uma dimensão. A relevância e/ou similaridade entre uma *query* e um documento é calculada por uma medida da distância vetorial entre esses vetores.
- Modelo probabilístico - é calculada a probabilidade de um documento corresponder a uma *query* usando uma abordagem probabilística.
- A abordagem do Lucene quanto aos modelos de pesquisa consiste em combinar o modelo booleano puro com o modelo de espaço vetorial, havendo a possibilidade de se decidir qual o modelo pretendido.

Por fim, o Lucene retorna os documentos encontrados de acordo com os critérios de pesquisa definidos pelo utilizador para serem renderizados e mostrados.

### 2.3.2 Apache Solr

O Apache Solr é uma popular plataforma de pesquisa de código aberto, baseada em Java e construída sobre o Apache Lucene (Grainger & Potter, 2014). Como participante na indústria há mais de uma década (lançado em 2004), oferece as suas capacidades de pesquisa de uma forma amiga do utilizador, sendo um produto maduro com uma forte e vasta comunidade. O Apache Solr caracteriza-se por ser (Grainger & Potter, 2014) (Apache, 2018):

- Altamente escalável;
- Tolerante a falhas;
- Extensível através de *plugins*;
- Capaz de lidar com largos volumes de dados;
- Otimizado para pesquisa;

- Capacidades avançadas de pesquisa por texto completo;
- Estrutura de indexação de documentos flexível;
- Disponibilizar APIs baseadas em REST;
- Disponibilizar os documentos para pesquisa quase imediatamente após terem sido indexados (quase em tempo real);

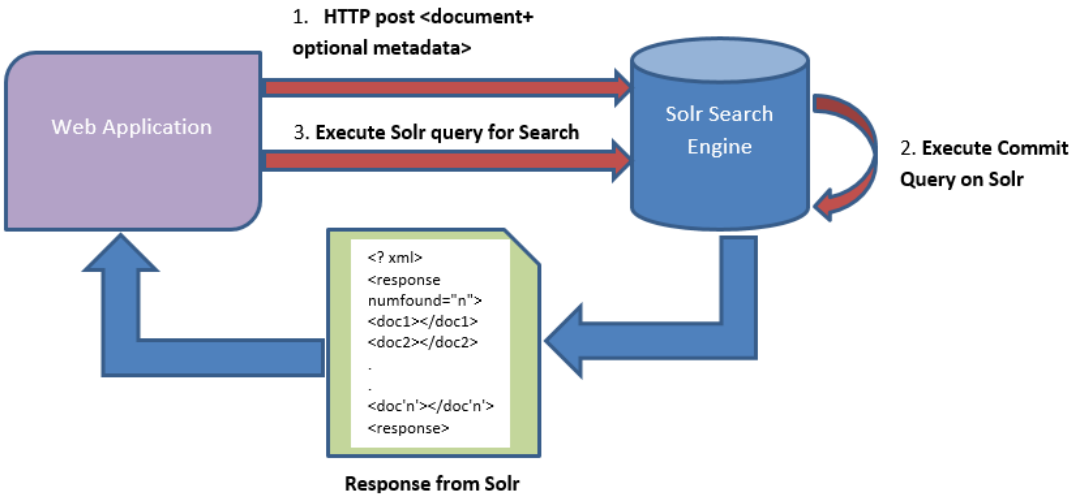



Figura 9 – Arquitetura do Solr para indexação e pesquisa de documentos (Morajkar, 2018).

Na Figura 9 encontra-se representado de forma resumida o processo de indexação e de pesquisa de documentos usando uma aplicação *web*. No processo de inserção de documentos no servidor Solr, a aplicação *web* estabelece comunicação com a plataforma através do protocolo HTTP, submetendo os documentos pretendidos para indexação. Os documentos a serem enviados para o Solr podem ser construídos ou carregados de forma manual, ou podem ser obtidos e construídos através de informação existente numa base de dados, apesar de na Figura 9 não estar representado este último caso. O Solr procede à importação e indexação desses documentos, ficando estes prontos para consulta no motor de pesquisa. Quando um utilizador inicia o processo de pesquisa de informação, a aplicação *web* converte esse pedido para uma *query* compreensível pelo Solr e submete essa *query* para o servidor. O Solr executa a *query* recebida e retorna a resposta obtida de acordo com a *query* efetuada.

### 2.3.3 Elasticsearch

O Elasticsearch, como o Apache Solr, foi desenvolvido com base no Apache Lucene e é um motor de pesquisa e de análise de código aberto. Este mecanismo de pesquisa distribuído foi desenvolvido por Shay Banon e lançado em 2010. O Elasticsearch estende a função do Apache

Lucene para armazenar, indexar e pesquisar dados de forma mais rápida, fácil e, tal como o nome indica, elástica (Hinman, Gheorghe, & Russo, 2015).

Acrescentando o que o Lucene fornece, o Elasticsearch oferece novas funcionalidades como o armazenamento em cache ou a análise em tempo real. Outra diferenciação comparativamente ao Lucene é a forma como se podem organizar os documentos: vários índices podem ser pesquisados separadamente ou em conjunto, e podem ser colocados diferentes tipos de documentos dentro de cada índice (Hinman, Gheorghe, & Russo, 2015).

A parte elástica desta biblioteca refere-se à capacidade de *clustering*: pode-se adicionar sempre mais servidores para aumentar a capacidade ou a tolerância a falhas. Contrariamente, também se pode remover facilmente os servidores do *cluster* para reduzir os custos se a carga de dados for menor (Hinman, Gheorghe, & Russo, 2015).

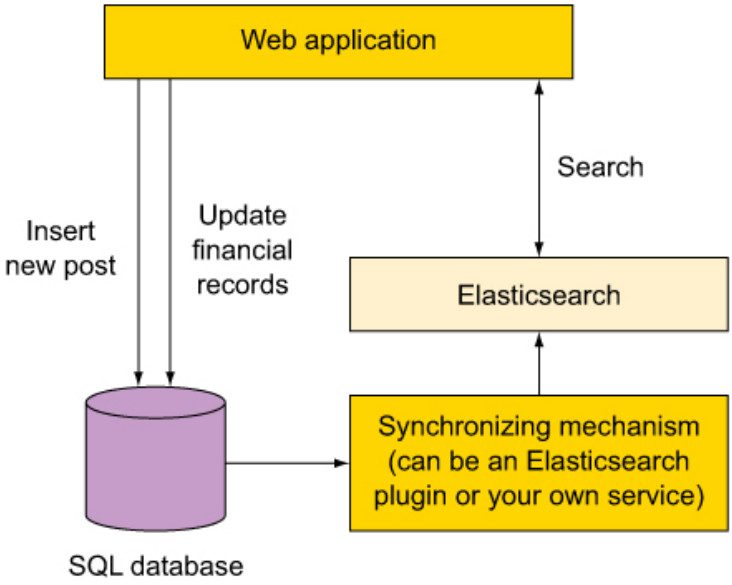



Figura 10 – Elasticsearch integrado com outro sistema de base de dados (Hinman, Gheorghe, & Russo, 2015).

Na Figura 10 apresenta-se a utilização do Elasticsearch num sistema com uma base de dados relacional. Quando é inserida uma nova informação ou editada uma já existente na base de dados, um mecanismo de sincronização, desenvolvido num serviço independente ou com um plugin do Elasticsearch, encarrega-se de sincronizar a informação atualizada, obtendo-a e transferindo-a para o Elasticsearch através da comunicação com a sua API. Essa informação é então recebida e indexada como um documento. Cabe, assim, ao mecanismo de sincronização manter o Elasticsearch atualizado com as últimas alterações na base de dados.

Quando um utilizador usa a funcionalidade de pesquisa na página web, a aplicação web consulta o Elasticsearch com uma query relativa aos critérios de pesquisa do utilizador. O Elasticsearch retorna vários documentos correspondentes aos critérios, classificando-os da maneira pretendida (Grainger & Potter, 2014).

Note-se que na Figura 10 encontra-se representada a utilização do Elasticsearch num sistema com base de dados, mas é possível, caso seja adequado, usá-lo como o único sistema de armazenamento de dados.

### 2.3.4 Comparação

Na Figura 11 e Figura 12 é apresentado respetivamente uma tabela e um gráfico com o ranking relativo à popularidade dos motores de pesquisa ao longo do tempo. Este ranking é baseado na medição da popularidade de um sistema usando os seguintes parâmetros (DB-Engines, 2018b):

- Número de menções do sistema em *websites* - medido como o número de resultados de *queries* usando os motores de pesquisa Google, Bing e Yandex;
- Interesse geral no sistema - frequência de pesquisas no Google Trends;
- Frequência de discussões técnicas sobre o sistema - medido verificando o número de perguntas relacionadas e o número de utilizadores interessados, nos *websites* de perguntas e respostas sobre tecnologias de informação, como o Stack Overflow e o DBA Stack Exchange;
- Número de ofertas de trabalho em que o sistema é mencionado - número de ofertas nos motores de pesquisa líderes no setor de oferta de emprego;
- Número de perfis em redes de profissionais em que o sistema é mencionado - são usadas redes profissionais populares internacionalmente, o LinkedIn e Upwork;
- Relevância nas redes sociais - medição do número de *tweets* em que o sistema é mencionado.

| Rank | | | DBMS | Database Model | Score | | |
|---|---|---|---|---|---|---|---|
| Dec 2018 | Nov 2018 | Dec 2017 | | | Dec 2018 | Nov 2018 | Dec 2017 |
| 1. | 1. | 1. | Elasticsearch | Search engine | 144.70 | +1.24 | +24.92 |
| 2. | 2. | ↑ 3. | Splunk | Search engine | 82.18 | +1.81 | +18.39 |
| 3. | 3. | ↓ 2. | Solr | Search engine | 61.35 | +0.47 | -4.95 |
| 4. | 4. | 4. | MarkLogic | Multi-model | 14.28 | +0.81 | +3.13 |
| 5. | 5. | 5. | Sphinx | Search engine | 7.82 | +0.46 | +1.79 |

Figura 11 – Tabela de *ranking* de popularidade motores de pesquisa (DB-Engines, 2018a).

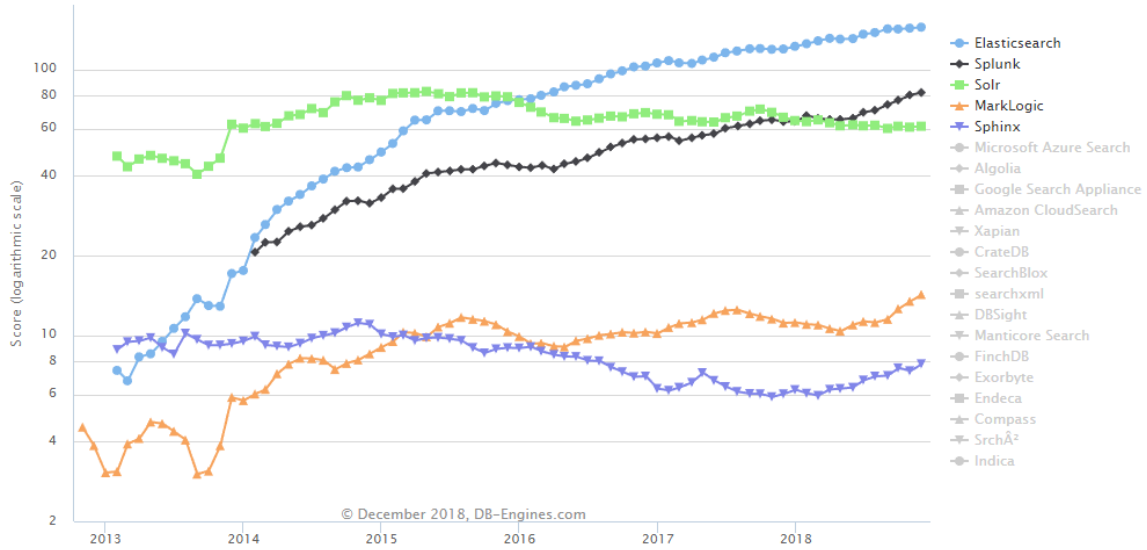



Figura 12 – Gráfico de *ranking* de popularidade motores de pesquisa(DB-Engines, 2018d).

Como podemos verificar na Figura 11 e Figura 12, o Elasticsearch e o Solr encontram-se nas primeiras posições relativamente à popularidade de motores de pesquisa. Ambos se mostram consistentemente como os dois motores de pesquisa mais populares ao longo dos anos, sendo que o Elasticsearch se tem cimentado no topo da tabela nos últimos tempos. O Apache Lucene não está incluído nesta comparação pois, dado que o Solr e o Elasticsearch são baseados no Lucene, estes estão logicamente mais desenvolvidos e abrangem um conjunto maior de funcionalidades e características.

Comparando o Elasticsearch com o Apache Solr, ambos providenciam funcionalidades semelhantes e características que evoluem a cada nova versão. Quando o Elasticsearch apareceu no mercado, tinha a grande vantagem de ser um modelo distribuído muito mais fácil de escalar comparativamente com os seus competidores, o que sugere a parte "*elastic*" do nome. Porém, com o passar do tempo, o Solr passou também a ter capacidade de fragmentação e escalonamento, o que tornou o argumento da distribuição discutível. Muitos programadores acham o Elasticsearch mais intuitivo, fácil e rápido de configurar e usar, enquanto o Solr apresenta um suporte melhor, com uma comunidade maior, bastante ativa e uma documentação abrangente e precisa (Kuc, 2017). Relativamente às suas APIs de acesso, o Elasticsearch tem o que se assemelha mais com REST APIs da era *Web* 2.0 mas, relativamente às suas Java APIs, o Solr apresenta uma API mais adequada para quem se encontra a trabalhar com esta linguagem, a Solrj (Tan, 2018).

Na Tabela 2 são comparadas diretamente algumas das características dos dois motores de pesquisa em análise.

Tabela 2 - Comparação de características entre Elasticsearch e Solr. Adaptado de DB-Engines (2018c), Kuc (2017)

| Nome | Elasticsearch | Solr |
|---|---|---|
| Modelo de base de dados principal | Motor de pesquisa | Motor de pesquisa |
| Modelo de base de dados secundário | Armazenamento de documentos | - |
| Desenvolvedor | Elastic | Apache Software Foundation |
| Lançamento inicial | 2010 | 2004 |
| Versão atual | 6.4.3, Novembro de 2018 | 7.5.0, Setembro de 2018 |
| Licença | Código aberto | Código aberto |
| Disponível apenas como serviço em nuvem | Não | Não |
| Linguagem de implementação | Java | Java |
| Comunidade e desenvolvedores | Entidade comercial e os seus colaboradores | Apache Software Foundation e suporte da comunidade |
| SO (sistemas operativos) dos servidores | Todos os SO com máquina virtual java | Todos os SO com máquina virtual java e *servlet container*[2] |
| Esquema de dados | Livre de esquema. Definições de tipos flexíveis. Depois de um tipo ser definido é persistido | Sim. Campos dinâmicos permitem a adição imediata de novos campos |
| Tipos de dados predefinidos (como *float* ou *date*) | Sim | Sim. Suporta tipos de dados personalizáveis e deteta os tipos automaticamente |
| Índices secundários | Sim | Sim |
| Suporte SQL | Linguagem de consulta semelhante a SQL | Não |
| Velocidade de pesquisa | Bom para dados que mudam rapidamente devido às caches por segmento | Melhor para dados estáticos devido ao sistema de cache e leitor não invertido |
| Funcionalidades de pesquisa de texto completo | Análise de linguagem baseada no Lucene, sugestão única de implementação da API, *highlighting rescoring* | Análise de linguagem com base no Lucene, múltiplas sugestões, verificadores |

[2] Servidor de aplicação que fornece os recursos necessários para executar Java Servlets mas não possui todas as especificações Java EE, sendo utilizado em aplicações mais simples. Um exemplo de um servlet container é o Tomcat.

| Nome | Elasticsearch | Solr |
|---|---|---|
| | | ortográficos, *rich highlighting support* |
| Controlo do líder do índice/coleção | Não é possível | Controlo do posicionamento do líder e possibilidade de rebalanceamento do líder para equilibrar a carga nos nós |
| Linguagem específica de domínio para a consulta (*query* DSL) | JSON | JSON, XML ou parâmetros URL |
| APIs e outros métodos de acesso | Java API. RESTful HTTP/JSON API | Java API. RESTful HTTP API |
| Linguagens de programação suportadas | .Net<br>Groovy<br>Java<br>Javascript<br>Perl<br>PHP<br>Python<br>Ruby<br>Clientes com origem na contribuição da comunidade | .Net<br>Erlang<br>Java<br>Javascript<br>Perl<br>PHP<br>Python<br>Ruby<br>Scala<br>Qualquer linguagem que suporte *sockets* e XML ou JSON |
| *Scripts* do lado servidor | Sim | Java plugins |
| Triggers | Sim, usando a função de percolação | Sim. Comandos configuráveis pelo utilizador são acionados em alterações no índice/coleção |
| Métodos de replicação | Sim | Nuvem/distribuído. Replicação mestre-escravo |
| Métodos de particionamento | Sharding | Sharding |
| Conceitos de consistência | Consistência eventual | Consistência eventual |
| Conceitos de transação | Não | Sim |
| Concorrência | Sim | Sim |
| Durabilidade (suporte para tornar os dados persistentes) | Sim | Sim |

Como se pode visualizar na tabela, o Elasticsearch apresenta algumas características que diferem do Solr, como por exemplo no suporte a linguagem SQL ou na linguagem específica de

domínio para a consulta dos dados, pelo que escolher entre eles pode depender unicamente do ambiente em que se trabalha e das funcionalidades e dos objetivos que se desejam alcançar. Normalmente, as funcionalidades necessárias para muitos casos de uso são abrangidas por ambos os motores, sendo difícil decidir qual das ferramentas se adapta melhor ao problema em causa, pelo que escolher entre o Elasticsearch e o Solr é, na opinião de alguns especialistas, uma questão de gosto e familiaridade (Hinman, Gheorghe, & Russo, 2015).

Na secção Análise e Avaliação é apresentada uma avaliação efetuada a estes dois motores de pesquisa e qual deles foi selecionado para fazer parte da solução a implementar.

## 2.4 Interface gráfica e funcionalidades de Bibliotecas Digitais

A interface gráfica é um dos componentes mais importantes de um sistema na medida em que é o componente do sistema visível pelo utilizador e, consequentemente, tem um grande impacto na sua opinião e decisão de usar o sistema ou não.

De forma a compreender como estão organizadas visualmente as principais páginas e funcionalidades de uma Biblioteca Digital moderna, nas secções seguintes são apresentados alguns dos estudos feitos sobre as tendências adotadas na interface gráfica de Bibliotecas Digitais e as suas funcionalidades. A partir desses estudos são efetuadas conclusões sobre os aspetos mais relevantes observados a ter em conta no desenvolvimento da solução. O foco deste estudo foram principalmente os itens que consistem na barra de navegação do *website*, a sua página inicial e a página de pesquisa. A forma como a barra de navegação é apresentada é importante para a usabilidade de qualquer *website*, uma vez que permite que o utilizador navegue e tenha acesso às diferentes páginas existentes, servindo normalmente como referência para as diferentes funcionalidades de um sistema. A página inicial é a página que causa o primeiro impacto na perspectiva do utilizador sobre o *website* que está a consultar. Por fim, a página de pesquisa avançada contribui muito para uma boa usabilidade dos *websites* que reuniem grandes quantidades de dados, auxiliando o utilizador na procura dos recursos pretendidos.

### 2.4.1 Biblioteca Digital Mundial

A Biblioteca Digital Mundial (*World Digital Library*) é uma plataforma *web* projetada pela Biblioteca do Congresso dos Estados Unidos da América e pela UNESCO e foi lançada em Abril de 2009, disponibilizando atualmente em formato multilíngue e gratuitamente importantes fontes de conteúdos provenientes de países e culturas de todo o mundo (World Digital Library, 2018).

#### 2.4.1.1 Barra de navegação

A barra de navegação está presente em todas as páginas do *website*, apresentando dois *layouts* diferentes dependendo da página aberta. Na página inicial e na página de pesquisa avançada a barra de navegação visível na Figura 13 não contém a área de pesquisa por texto, uma vez que esta já aparece noutra localização da página.


Figura 13 - Barra de navegação na Homepage da Biblioteca Digital Mundial (Biblioteca Mundial, 2018b).

Em todas as restantes páginas do *website* a barra de navegação contém a área de pesquisa por texto, como se pode visualizar na Figura 14.

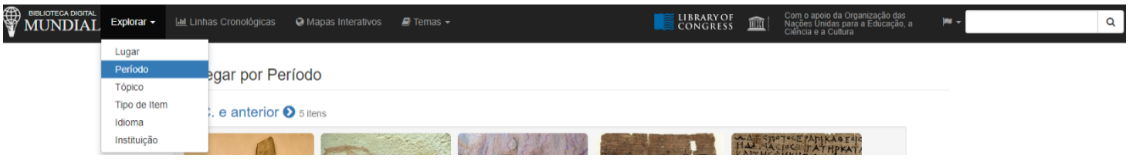



Figura 14 – Barra de navegação da Biblioteca Digital Mundial com área de pesquisa (Biblioteca Mundial, 2018a)

A barra de navegação permite explorar itens por lugar, período, tópico, tipo de item (como vídeo e imagem), idioma e instituição. Para além disso, destaca-se ainda o atalho para visualizar as linhas cronológicas e mapas interativos existentes. As linhas cronológicas consistem em apresentar todos os itens de um determinado tema por ordem cronológica, e os mapas interativos demonstram pontos no mapa a que estão associados itens da biblioteca.

#### 2.4.1.2 Página Inicial

Na página inicial, exibida na Figura 15, são apresentados num *slideshow* vários itens de informação que o utilizador pode selecionar para visualizar a informação desse item. Há também um duplo espaço de quatro posições, um deles para exibição de quatro instituições a que estão associados diversos itens e outro para exibir os quatro itens adicionados recentemente. Para além disso, na parte inferior da página é apresentada uma área onde são exibidos itens de destaque e uma pequena parte do seu conteúdo.

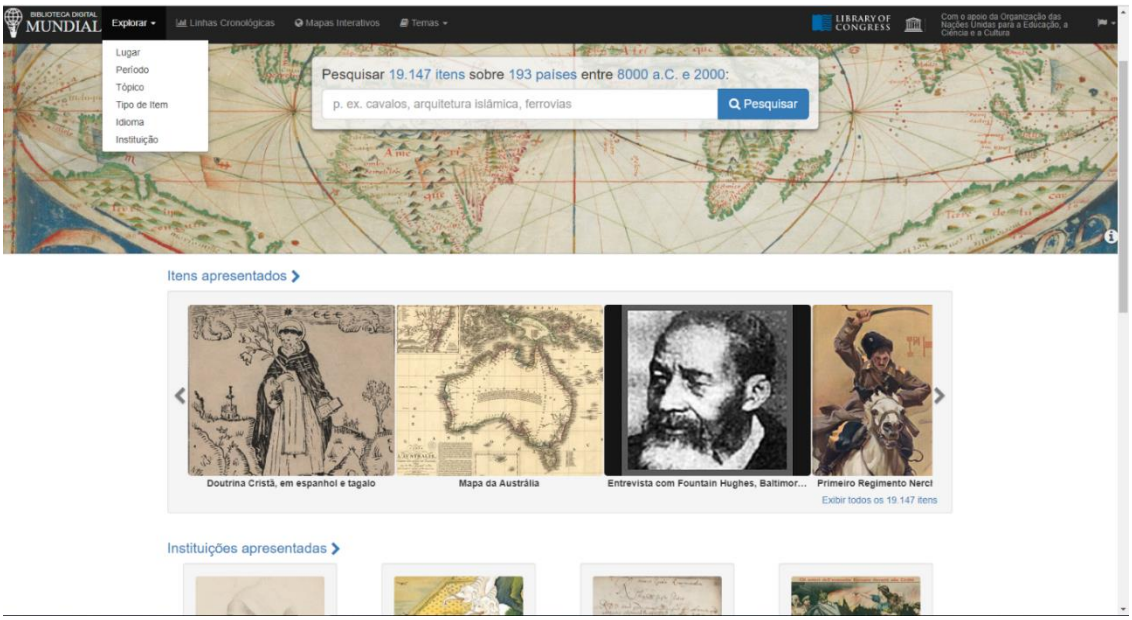



Figura 15 – Página Inicial da Biblioteca Digital Mundial (Biblioteca Mundial, 2018b)

Esta página é também constituída por uma barra de pesquisa de texto bem destacada, onde o utilizador pode inserir o texto que desejar para inicializar a sua pesquisa.

#### 2.4.1.3 Página Resultados da Pesquisa

A página de resultados da pesquisa apresenta os resultados da pesquisa efetuada e disponibiliza funções avançadas de pesquisa para filtrar os resultados encontrados. Caso um utilizador tenha feito uma pesquisa de texto na área de pesquisa da página inicial ou da barra de navegação, é reencaminhado para esta página onde são apresentados os respetivos resultados. Caso o utilizador tenha selecionado a opção de pesquisa sem ter inserido nenhum texto, é na mesma redirecionado para esta página para efetuar a sua pesquisa com funções avançadas. Na Figura 16 é apresentada a página de pesquisa da Biblioteca em estudo.

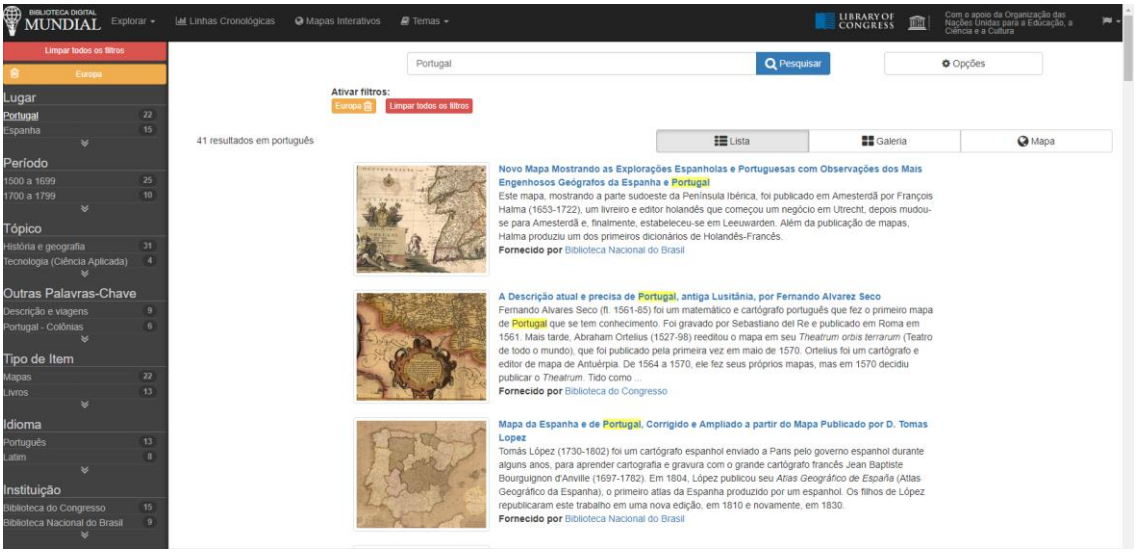



Figura 16 – Página de resultados de pesquisa da Biblioteca Digital Mundial (Biblioteca Mundial, 2018c)

Esta página permite a seleção de determinados filtros, como a escolha do lugar, período, tópico, tipo de item, idioma e instituição, bem como a escolha de palavras-chave relevantes. Estes filtros, juntamente com o texto escrito na área de pesquisa aberta, refinam a pesquisa restringindo os resultados obtidos. É possível ver os filtros ativados e desativá-los individualmente ou limpar todos os filtros ativados. À medida que cada filtro é ativado ou desativado são atualizados os resultados apresentados, que podem ser visualizados em forma de lista ou grelha. A paginação dos resultados é efetuada através do *scroll* para a parte inferior da página.

### 2.4.2 Europeana

A Europeana é uma Biblioteca multimédia desenvolvida por países da União Europeia e lançada em 2008 com o objetivo de partilhar com todas as pessoas o património cultural da Europa (Comissão Europeia, 2010).

#### 2.4.2.1 Barra de navegação

A barra de navegação do *website* da Europeana mantém a sua organização em todas as páginas existentes. Esta barra é constituída por quatro menus principais: coleções, explorar, exposições e blogue. Estes menus permitem, respetivamente, navegar por coleção (Figura 17), explorar conteúdos por diferentes tipos de parâmetros (Figura 18), como fontes, tópicos, período, entre outros, navegar por exposições disponíveis e consultar *posts* do blogue da Europeana.

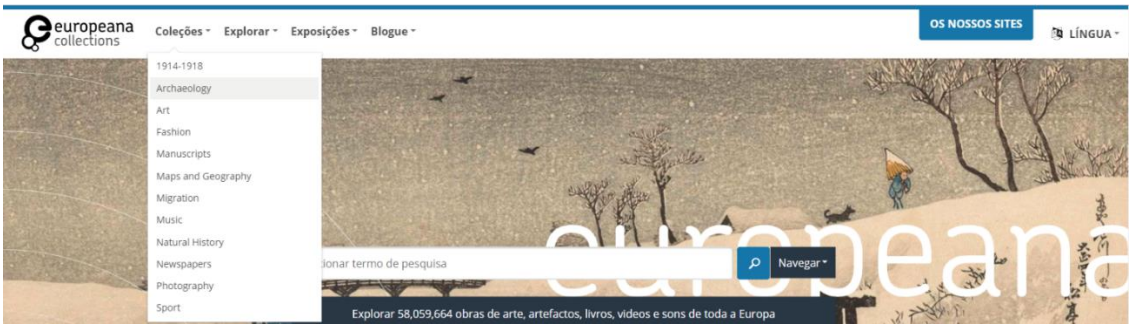



Figura 17 – Barra de navegação menu Coleções da Europeana (Europeana, 2018d)


Figura 18 – Barra de navegação menu Explorar da Europeana (Europeana, 2018d)

#### 2.4.2.2 Página Inicial

A página inicial, apresentada na Figura 20, tem uma área de destaque para pesquisa de texto livre, onde o utilizador pode inserir os termos de pesquisa que desejar e selecionar o tipo de formato de conteúdo que deseja associar a esses termos (imagem, som, texto, vídeo, objetos 3D ou todos). Esta área de pesquisa existe na maioria das páginas do *website*, com a diferença de que à medida que se vai restringindo a área de navegação (por exemplo ao navegar numa determinada coleção), o termo associado a essa área é adicionado automaticamente como filtro de pesquisa na área de pesquisa. Nas páginas em que não existe esta área de texto livre de pesquisa (na navegação por exploração), é disponibilizada uma lista com as coleções existentes de forma a filtrar a informação apresentada pela coleção selecionada, como se pode visualizar na Figura 19.

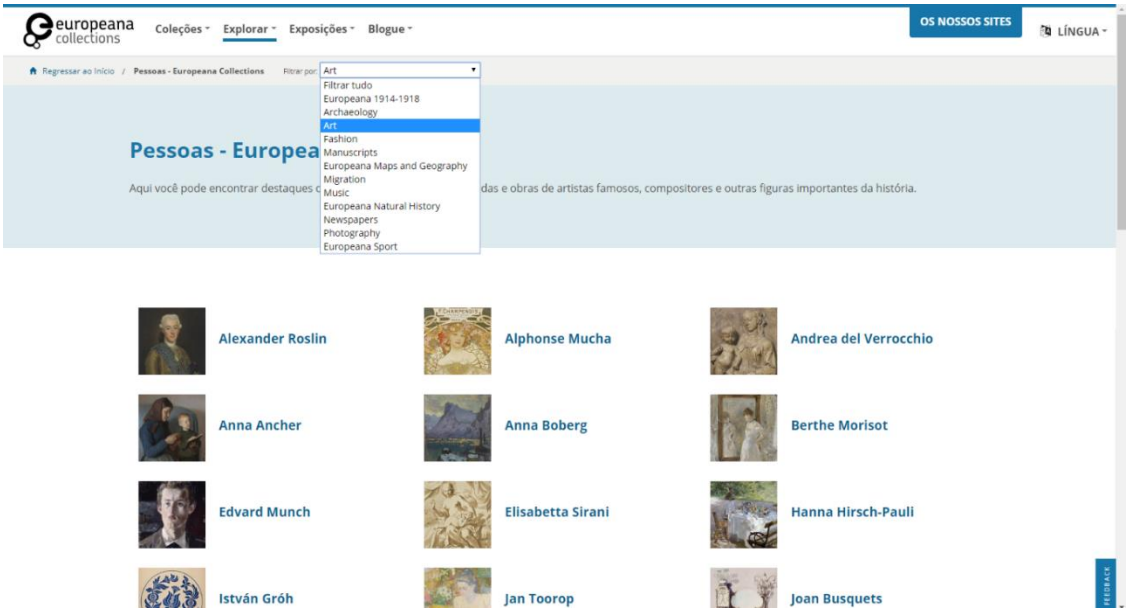



Figura 19 – Filtro de informação da página acedida através do menu Explorar (Europeana, 2018e)

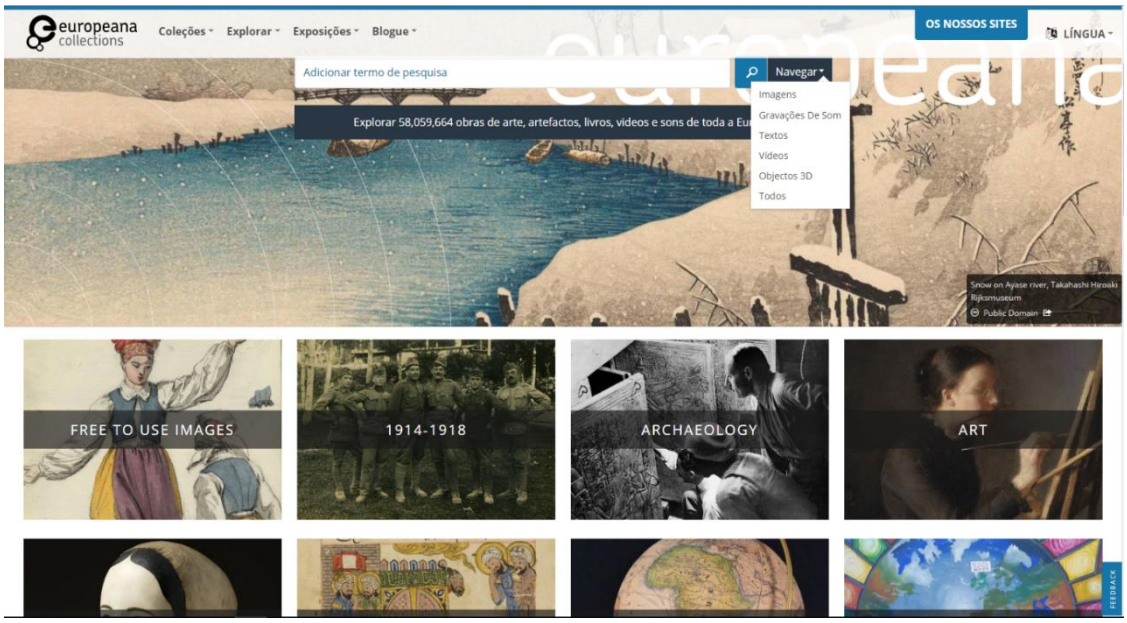



Figura 20 - Homepage da Europeana (Europeana, 2018d)

Na página inicial são também apresentadas em destaque, sobre a forma de grelha, as coleções existentes na Biblioteca e são expostas no fim da página as três últimas notícias publicadas no blog da Europeana.

#### 2.4.2.3 Página Resultados da Pesquisa

A página de resultados da pesquisa desta Biblioteca pode ser acedida através de duas formas: usando diretamente a área de pesquisa de texto livre, ou através da seleção de uma coleção e

de uma das opções de exploração dessa coleção. Na Figura 21 é apresentada a página de pesquisa da Europeana.

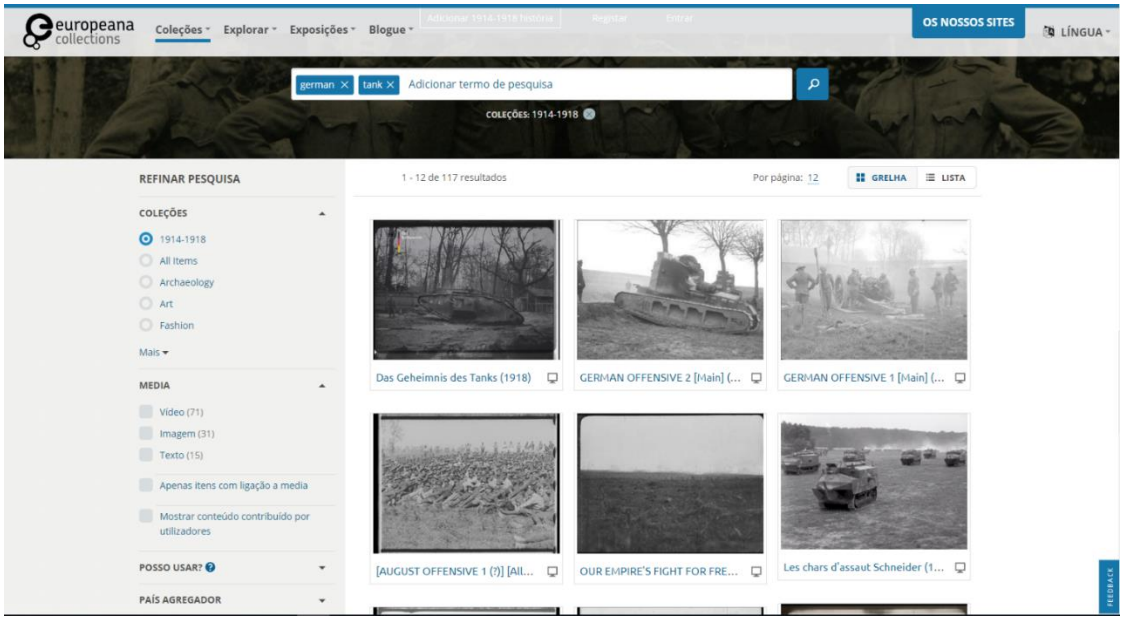



Figura 21 – Página de Resultados de Pesquisa da Europeana (Europeana, 2018f)

A filtração dos resultados da pesquisa pode ser efetuada inserindo na área de texto livre vários termos de pesquisa e/ou refinando a pesquisa através da seleção de diversos filtros: coleção, tipo de formato do item (vídeo, imagem, texto), estado de direitos de autores, país agregador, língua do conteúdo, agregador do item à Biblioteca e Instituição detentora do item. Os resultados podem ser visualizados em grelha ou em lista e a paginação dos resultados é constituída por 12 resultados por página, existindo a opção de modificar este valor para 24, 48 ou 96 resultados por página.

#### *2.4.2.4* Submissão de itens

Na Europeana é possível um utilizador registar-se como um contribuidor e submeter o seu contributo para revisão e aprovação de especialistas da Biblioteca. À data do estudo efetuado, esta Biblioteca apenas permite submeter contributos para uma coleção, no âmbito de um projeto denominado Europeana 1914-1918 que tem como objetivo recolher documentos e objetos relacionados com a grande guerra e com as pessoas envolvidas e afetadas por ela (Europeana, 2018c). Para isso, o utilizador contribuidor deve selecionar a coleção 1914-1918 e a opção de adicionar história, ou então dirigir-se à sua conta e adicionar uma nova história. A partir daí, é requisitado a preencher um conjunto de informações divididas em três passos: o primeiro passo relaciona-se com a história a submeter, como o título, uma descrição da história, uma descrição resumida dos itens digitais a adicionar, a língua em que a história está a ser escrita, os participantes da história, o momento temporal e espacial da história e palavras-chave relacionadas com a história; o segundo passo, apresentado na Figura 22, consiste em adicionar

itens (ficheiros) à história. O contribuidor deve selecionar o(s) ficheiro(s) a submeter e preencher as informações requisitadas sobre o(s) item, como o seu título, nome e apelido do seu criador, número específico da página (caso seja um item com muitas páginas), lado (caso seja um item com dois lados, como um postal), a sua descrição, língua, tipo (fotografia, texto digitalizado, vídeo, áudio), data e local de criação e palavras-chave associadas; o terceiro e último passo consiste em dar por concluída a submissão ao confirmar a intenção de submeter a história. Uma vez submetida, a contribuição será revista por um catalogador da Europeana e disponível ao público caso seja aceite (Europeana, 2018c).

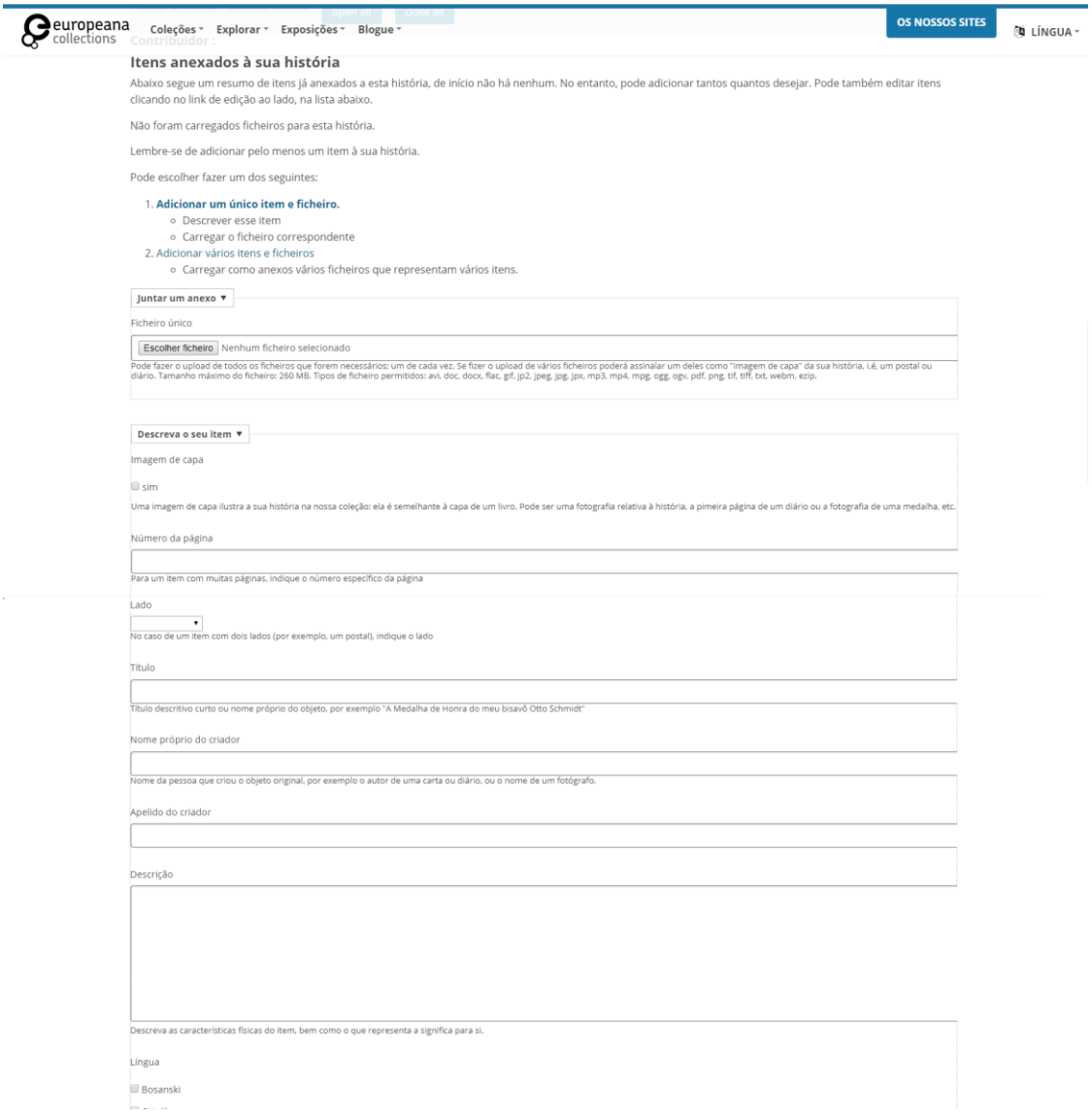



Figura 22 – Página de submissão de itens a uma história (Europeana, 2018b).

### 2.4.2.5 Página da conta do contribuidor

Esta página, visível na Figura 23, permite que cada utilizador registado no sistema possa visualizar e editar os seus dados de início de sessão e os seus dados pessoais.

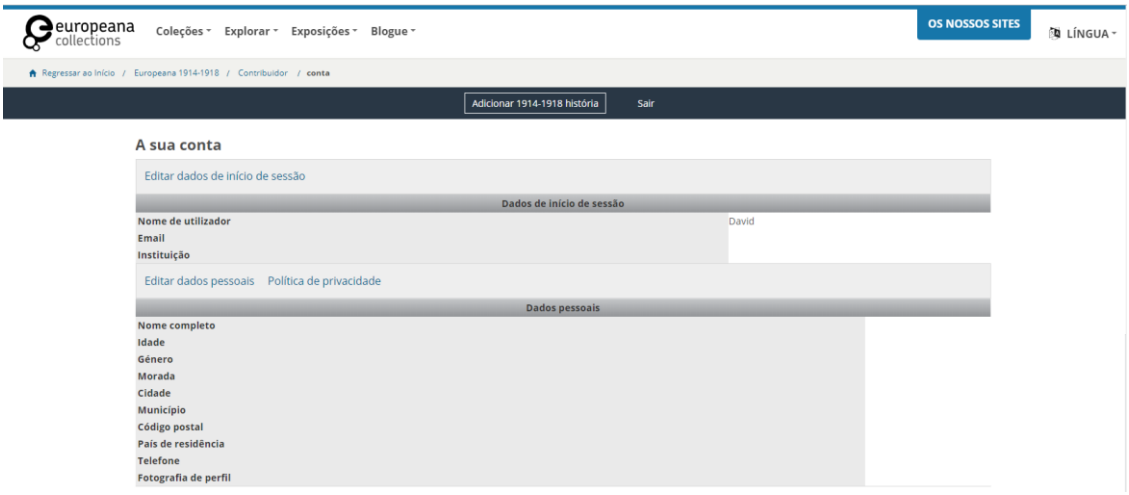



Figura 23 – Página da conta de utilizador da Europeana (Europeana, 2018a).

Como se pode constatar na figura anterior (Figura 23), os dados pessoais consistem no nome completo, idade, género, morada, cidade, município, código postal, país de residência, telefone e fotografia de perfil. A partir desta página, o utilizador pode selecionar a opção de adicionar uma história, sendo redirecionado para a página apresentada na Figura 24, onde pode então escolher adicionar uma nova história e/ou visualizar as histórias que submeteu anteriormente e ter a hipótese de, caso ainda não tenha sido submetida a história, editá-la ou eliminá-la.


Figura 24 – Página de informação das histórias do contribuidor (Europeana, 2018c).

### 2.4.3 Conclusões

Através do estudo efetuado conclui-se alguns pontos fundamentais:

- A barra de navegação de uma Biblioteca deve estar sempre presente e ser de fácil usabilidade e acima de tudo útil, devendo estar bem organizada de forma a que seja possível navegar no *website* e explorar os itens de forma intuitiva através da exposição das categorias de conteúdos existentes;

- Os itens das Bibliotecas encontram-se bem organizados e divididos através da sua associação a determinadas categorias/coleções/tipos de conteúdo;

- A página inicial das Bibliotecas apresenta os conteúdos de forma organizada através da sua categorização, apresentando, por exemplo, os conteúdos em destaque, conteúdos adicionados recentemente, ou pelas suas categorias (por exemplo arte, desporto, música);

- As Bibliotecas estudadas apresentam uma página de pesquisa avançada, onde são apresentados os resultados de uma pesquisa efetuada e onde é possível refinar a pesquisa através da escolha de diversos filtros. Os resultados podem ser vistos em forma de grelha ou lista;

- Durante a navegação no *website* normalmente existe uma área de pesquisa de texto livre para auxiliar o utilizador a encontrar o que deseja. Esta área localiza-se regularmente na barra de navegação ou numa área de destaque da página e serve como um atalho para a página de pesquisa avançada onde são exibidos os resultados da pesquisa.

# 3 Análise de Valor

Nesta secção é apresentado o processo de análise de valor relativo ao produto desenvolvido, bem como os métodos e técnicas utilizadas para identificar o valor do produto e as funcionalidades principais que geram verdadeiro valor para o cliente. O modelo adotado para efetuar esta análise de valor é o *Value Engineering* (VE). Este modelo é simular à abordagem *Value Analysis* (VA), com a principal diferença de se destinar a novos produtos ou serviços, operando-se assim num ambiente mais incerto e com menor informação disponível para efetuar decisões (Rich & Holweg, 2000).

O VA pode ser definido como um processo sistemático, formal e organizado de análise e avaliação de forma a compreender se um produto cumpre os requisitos exigidos pelo cliente com o menor custo possível correspondente às necessidades de desempenho e confiabilidade (Rich & Holweg, 2000).

O foco principal da abordagem de análise de valor é a gestão da funcionalidade (Rich & Holweg, 2000), seguindo um processo composto por diferentes técnicas para identificar as funções principais que realmente trazem valor para o cliente, compreendendo quais os aspetos tangíveis e intangíveis que são valorizados pelos consumidores do produto e quais os aspetos que não interessam. Como referido anteriormente, caso a análise recaia sobre novos produtos, deve-se usar a abordagem VE que aplica os mesmos princípios e muitas das técnicas do VA.

O processo de análise de valor é constituído por cinco fases, representadas na Figura 25: Orientação, Análise Funcional (inclui também a fase de Identificação Funcional), Criação de Alternativas, Análise e Avaliação, Implementação.

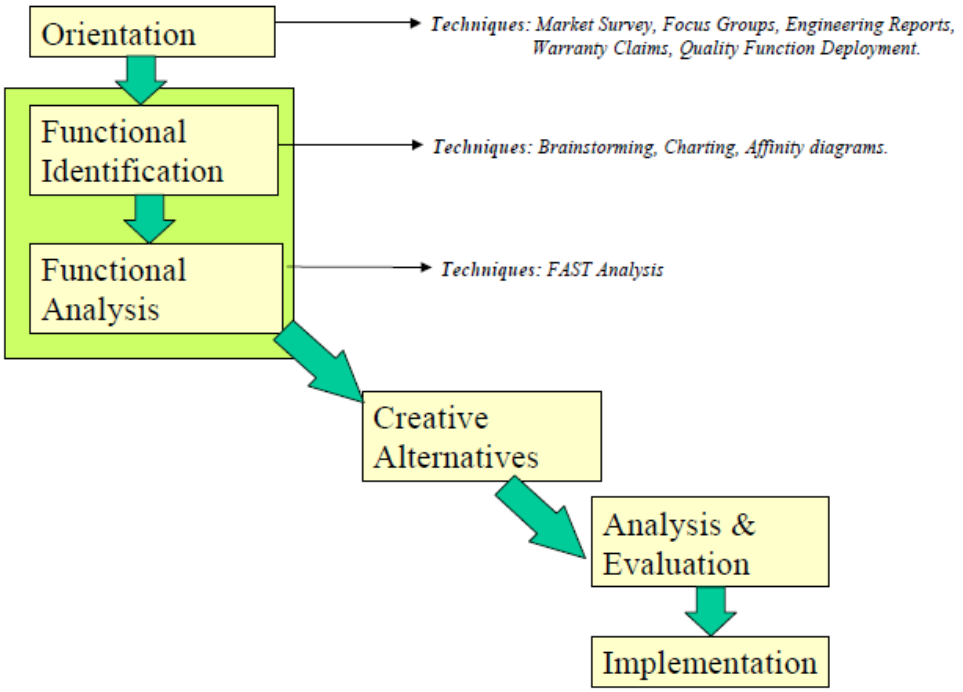



Figura 25 – Processo de Análise de Valor (Rich & Holweg, 2000)

A etapa de Orientação tem como objetivo a escolha e descrição do produto ou serviço que irá ser desenvolvido e alvo de estudo neste processo de análise. A fase de Análise Funcional foca-se na identificação, descrição e análise das funcionalidades do produto que fornecem maior valor para o cliente. A etapa de Criação de Alternativas consiste na identificação e levantamento de ideias de forma a reunir possíveis soluções para o produto. A quarta fase, de Análise e Avaliação, serve para avaliar o custo e o valor de cada função (Rich & Holweg, 2000) e comparar as alternativas previamente definidas, baseado num conjunto de critérios para concluir qual a mais adequada para o produto a desenvolver. A Implementação é a última etapa do processo de análise de valor e tem como objetivo a descrição do desenvolvimento do produto.

## 3.1 Orientação

Para a primeira etapa do processo de análise de valor foi aplicado o modelo *Fuzzy Front End* (FFE) que faz parte do processo de inovação (Koen, et al., 2002). Este processo é constituído por três fases distintas: o FFE, que consiste em atividades normalmente caóticas, imprevisíveis e não estruturadas para definir a base do produto; o *New Product Development* (NPD), mais disciplinado, constituído por ações formais e bem estruturadas aplicadas no desenvolvimento do novo produto; a comercialização é a última fase do processo de inovação e pode ser apresentada como o estudo das atividades de marketing para a divulgação e lançamento do produto (Koen, et al., 2002).

De forma a descrever os componentes do *Fuzzy Front End* numa notação e terminologia comum e compreensível para todas as pessoas, Peter Koen desenvolveu o *New Concept Development Model* (NCD), apresentado na Figura 26.

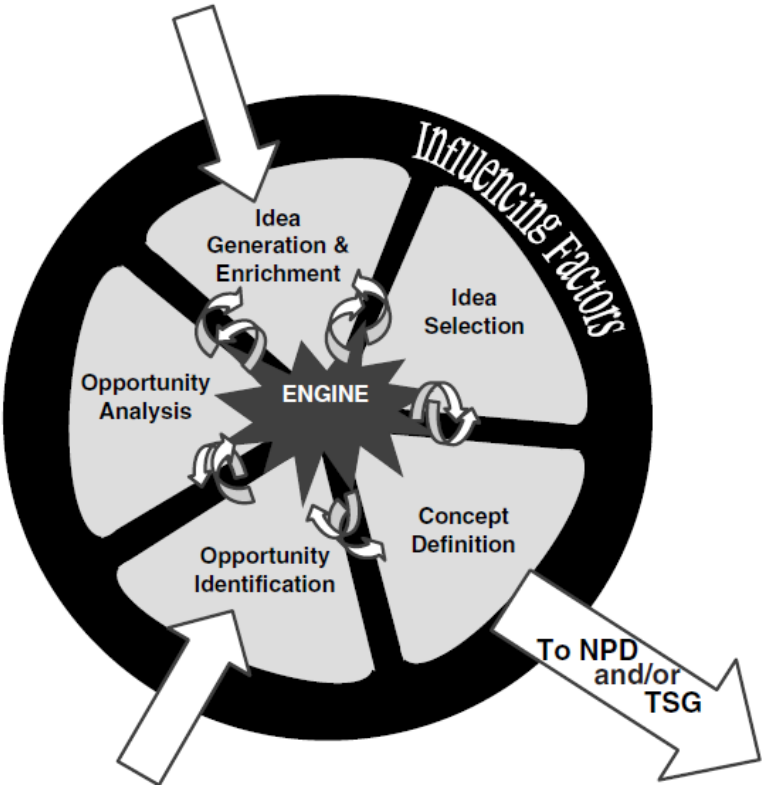



Figura 26 – *New Concept Development Model* (Koen, et al., 2002)

O NCD é constituído por três partes principais:

- Fatores de Influência;
- Motor;
- Os cinco elementos do FFE - Identificação de oportunidade, Análise de oportunidade, Geração e enriquecimento de ideias, Seleção de ideias e Definição de conceito.

O modelo NCD apresenta uma forma circular com o objetivo de representar um fluxo de circulação e iteração entre todos os cinco elementos chave do NCD. De seguida são apresentados os elementos deste modelo no contexto deste projeto.

### 3.1.1 Fatores de influência

Os fatores de influência são fatores que afetam o processo de inovação e desenvolvimento de um produto. Estes fatores consistem nas capacidades organizacionais da empresa, estratégia de negócio e influências externas, como a política, a alteração, remoção ou adição de leis, o perfil dos clientes e as suas necessidades, canais de distribuição existentes, a evolução da concorrência e também o estado da ciência e das tecnologias existentes (Koen, et al., 2002). Estes fatores, como já referido, afetam todo o processo de inovação e podem ser incontroláveis pela organização.

Os maiores fatores de influência a considerar no contexto deste projeto são as capacidades organizacionais da empresa, os seus clientes e concorrência, e o estado das tecnologias e ferramentas existentes no mercado que melhor se enquadram nas necessidades do projeto.

Sendo uma empresa inserida na área da saúde, a ALERT recolhe ativamente informações de fontes externas através de conferências, notícias e outros dados relacionados com essa área de forma a estar ocorrente das mudanças, problemas e oportunidades que possam surgir neste mercado e tenham impacto nos seus produtos. Quando há fatores de influência externos que provoquem a adição ou modificação relevante de produtos da ALERT, procede-se à criação ou alteração de documentos e/ou vídeos que descrevam e expliquem as mudanças efetuadas para aprendizagem e conhecimento das mesmas. Assim, estes fatores de influência externos têm impacto no contexto deste projeto uma vez que esses novos conteúdos poderão ser adicionados, publicados e visualizados usando o produto deste projeto. Para além disso, a confiabilidade da empresa depende da sua capacidade de se adaptar às alterações do mercado e às necessidades dos clientes, respeitando sempre nos seus produtos os padrões e regras de saúde impostas.

Outro grande fator de influência são os clientes da ALERT, visto que a subsistência e crescimento da organização depende da sua satisfação e adesão aos produtos disponibilizados. Assim, é necessário estar em contacto constante com os clientes de forma a atender às suas necessidades e determinar de que forma o produto pode ser valorizado.

Para além da preocupação com os clientes, a ALERT está atenta aos seus concorrentes e à competição existente no mercado. Sendo um mercado que lida diretamente com a saúde dos seres humanos, nem todos os mercados confiam em sistemas de informação para efetuar a gestão da saúde dos indivíduos, o que torna esta área bastante competitiva. Dado que os padrões e exigências dos clientes à adoção de um sistema de gestão de saúde é bastante alta, manter um conhecimento sobre os movimentos e evoluções dos concorrentes é um fator importante para o valor do negócio.

Por fim, as tecnologias existentes no mercado são outro fator de influência para a organização. As tecnologias e ferramentas usadas no produto são de extrema importância pois, para além do impacto a curto prazo, podem ter impacto a longo prazo no futuro do sistema, afetando a

sua arquitetura e funcionalidades, a sua confiabilidade e durabilidade sem necessidade de grandes mudanças e manutenções.

### 3.1.2 Motor

O Motor é o que potencia e conduz os cinco elementos principais do NCD. Representa a liderança, cultura e estratégia de negócio da organização.

A proficiência é o que distingue as empresas altamente inovadoras das menos inovadoras (Koen, et al., 2002). Considerando a área de atuação da ALERT, a sua intenção de melhorar a qualidade de vida humana e, principalmente, a sua missão, visão e valores, a organização procura inovar na criação de produtos de qualidade. Para isso, a ALERT é constituída por uma cadeia organizada de equipas que se comprometem com a excelência, talhadas para tarefas específicas e cooperando entre si continuamente para garantir o cumprimento dos padrões de qualidade exigidos nos produtos, de forma a satisfazer as necessidades dos clientes e procurando cumprir sempre o objetivo da organização: melhorar a saúde e prolongar a vida, alcançar rentabilidade para benefício da sociedade e inspirar outros para a excelência através do seu exemplo (ALERT, 2018).

### 3.1.3 Identificação da oportunidade

Este elemento do NCD representa a etapa em que são identificadas oportunidades que interessem à organização para melhorar o valor dos seus produtos, dando especial atenção às tendências de mercado na área da saúde e à evolução tecnológica (Koen, et al., 2002).

Tendo em conta a dimensão da empresa e dos seus produtos, existe um número elevado de recursos de conhecimento/aprendizagem (vídeos, documentos, apresentações) sobre os produtos criados e de como interagir com as suas funcionalidades. A visualização destes conteúdos não tem o apoio de nenhuma plataforma, pelo que a sua procura é efetuada usando diretamente o sistema de ficheiros da rede da organização. O produto de eLearning da ALERT possui uma área de visualização do conteúdo dos cursos de aprendizagem realizados. Ao verificar a utilidade desta funcionalidade para os seus utilizadores e a sua satisfação por encontrarem facilmente e de forma prática num local os conteúdos que desejavam consultar relativamente aos cursos de aprendizagem, foi identificada uma nova oportunidade de desenvolver um produto independente, mais abrangente e com mais funcionalidades dedicado exclusivamente à gestão e visualização de conteúdos de diferentes formatos multimédia (Biblioteca Multimédia). Para além disso, um dos aspetos fundamentais do eLearning é a qualidade e quantidade de conteúdo e, neste sentido, as Bibliotecas Multimédia apoiam o ensino e a aprendizagem não só por desempenharem um papel em partilhar, preservar e organizar conteúdo, mas também pelo aspeto colaborativo de reunirem diferentes pessoas com diferentes perspetivas e ideias. Todos estes fatores aliados à possível redução de custos da empresa na educação e melhoramento das *skills* dos seus colaboradores com a fácil

acessibilidade e centralização dos conteúdos contribuíram para ter em consideração esta oportunidade.

Por todos estes motivos e pela verificação em ambiente real do sucesso que a construção de uma Biblioteca Digital teve na Airbus, permitindo poupar muito dinheiro e recursos (Annesley, 2017), fez com que a identificação desta oportunidade ganhasse mais relevo e importância para a ALERT.

### 3.1.4 Análise da oportunidade

A análise de oportunidade consiste em analisar uma oportunidade identificada no elemento anterior para averiguar o potencial e o valor dessa oportunidade e compreender se pode ser traduzida numa oportunidade de negócio.

O *perceived value* sobre o produto baseia-se na perspetiva global do consumidor sobre a sua utilidade, baseado nas perceções do que é recebido e do que é dado (Zeithaml, 1988). De forma a compreender o valor que um produto ou uma oportunidade identificada possa trazer para o cliente, Woodall definiu uma lista de indicadores (presente no Anexo A) onde apresenta um conjunto de benefícios e sacrifícios que podem ser aplicados à oportunidade identificada (Woodall, 2003).

Tendo em consideração esses indicadores, procedeu-se à identificação dos benefícios e sacrifícios (Tabela 3) tendo em conta o produto, o serviço e a relação entre o cliente e o fornecedor. No contexto deste projeto, os principais clientes são os colaboradores internos da empresa e os utilizadores do produto de eLearning.

Tabela 3 - Benefícios e Sacrifícios

| | Produto | Serviço | Relação |
|---|---|---|---|
| Benefícios | Qualidade de desempenho;<br>Redução de tempo despendido na gestão e procura de conteúdos;<br>Melhor experiência de aprendizagem para o utilizador;<br>Colaboradores melhoram o seu desempenho e mantêm as suas habilidades atualizadas;<br>Utilidade e usabilidade. | Disponibilidade permanente para consulta de conteúdo (confiabilidade);<br>Assistência na procura de conteúdo;<br>Rápida resposta na procura de conteúdo;<br>Flexibilidade no armazenamento de conteúdo. | Supervisão;<br>Assistência;<br>Confiança. |
| Sacrifícios | Custos de tempo de pesquisa para o desenvolvimento;<br>Custos de tempo de implementação;<br>Custos de instalação. | Custos de manutenção. | Custos de treino/formação dos colaboradores no uso do serviço. |

Considerando o produto e o seu valor para o cliente, o principal benefício consiste na utilidade que este produto trará para os clientes, uma vez que facilitará bastante o processo de visualização de conteúdos, bem como o fato de conter diversas fontes de informação que podem auxiliar os clientes na sua aprendizagem, proporcionando-lhes assim uma melhor experiência. Todos estes benefícios têm consequentemente um impacto positivo na empresa, reduzindo os seus custos no desenvolvimento das *skills* dos seus colaboradores, uma vez que este produto permite chegar a todos eles de forma fácil e organizada, mantendo-os atualizados sobre novos conhecimentos a adquirir. Sobre os sacrifícios, a implantação do produto no servidor de forma a estar pronto a ser utilizado tem como sacrifício os custos de instalação e, para além disso, verificam-se também os pontos negativos do tempo e recursos despendidos na pesquisa e no desenvolvimento do produto.

Relativamente ao serviço fornecido pelo produto, com a constante alteração e criação de novos conteúdos na ALERT e a necessidade de efetuar a sua consulta, é importante que exista uma área para efetuar operações de gestão e visualização de conteúdos, facilitando e tornando mais prático todo o processo. Com este produto vai ser disponibilizado um serviço que proporciona diversos benefícios para os seus clientes: disponibilidade 24h do serviço; assistência automatizada na procura de conteúdo; redução no tempo de procura de conteúdos; flexibilidade no armazenamento do conteúdo, permitindo que sejam armazenados diferentes tipos de formatos de conteúdo multimédia. Relativamente aos sacrifícios, a necessidade de manter o serviço disponível permanentemente e de possivelmente efetuar evoluções ao longo do tempo traduz-se em custos de manutenção.

Por último, na relação com a organização, os maiores benefícios que proporcionam valor para os clientes são a supervisão e assistência contínua no produto e serviço prestado, estabelecendo uma relação de confiança entre as duas partes. Como sacrifício, há a necessidade de despender algum tempo na formação dos colaboradores sobre o novo produto, uma vez que a ALERT segue a política de dar formação a todos os seus colaboradores quando algum novo produto é adicionado.

Em suma, a proposta de valor consiste em disponibilizar e simplificar o acesso e gestão de conteúdos multimédia de aprendizagem, contribuindo para o aumento do conhecimento e desempenho profissional, de forma prática e rápida com um produto de qualidade e design intuitivo.

### 3.1.5 Ideia

Atentando a todas as fases anteriores do processo NCD, a ideia deste projeto consiste em ter um local exclusivo de fácil acessibilidade para efetuar a adição e consulta de conteúdo de uma forma rápida e prática com o auxílio do sistema (tecnologias de pesquisa) a partir de qualquer local com acesso à internet, centralizando as fontes de informação num só sítio e beneficiando

os utilizadores promovendo a sua aprendizagem com diversos conteúdos. Assim procura-se manter os utilizadores ocorrentes do funcionamento dos produtos da ALERT e das suas mais recentes alterações, acompanhando o estado atual desses produtos que são fundamentais para melhorar e prolongar a saúde dos utentes.

### 3.1.6 Conceito

Considerando os elementos do NCD atrás analisados, o conceito do projeto pode ser descrito como o desenvolvimento de uma plataforma *web* multimédia com o objetivo de fornecer operações de gestão e visualização de diferentes formatos de conteúdos de aprendizagem sobre a ALERT e os seus produtos, complementando as informações dos cursos presentes no ALERT® eLEARNING, disponibilizando e partilhando esses conteúdos e permitindo a interação e colaboração entre utilizadores.

## 3.2 Análise Funcional

O primeiro passo do processo de Análise Funcional consiste em identificar as principais funcionalidades do produto (Rich & Holweg, 2000). A descrição detalhada destas funções pode ser visualizada na secção 4.1.1. Assim, uma vez que o produto terá um número considerado de funcionalidades, são apresentadas apenas as funções fundamentais:

- Autenticação e autorização;
- Gerir conteúdo (visualizar, adicionar, remover e editar);
- Pesquisa de conteúdo (visualizar e filtrar resultados da pesquisa);

Após a indicação das principais funções do produto é efetuada uma identificação da prioridade e importância dessas funções através de um *Pairwise Comparison*. Este método permite definir quais as funções mais importantes contabilizando os valores de importância atribuídos na comparação direta entre funcionalidades (cada funcionalidade é comparada diretamente com todas as outras). Existe uma escala de três valores de importância definidos: *minor* (1 ponto), *medium* (2 pontos) e *major* (3 pontos). Depois de todos os pares de funções terem sido comparados, são somados os pontos de cada função resultando no resultado final que define o seu valor final de importância (Rich & Holweg, 2000). Na Figura 27 é apresentado o *Pairwise Comparison.*

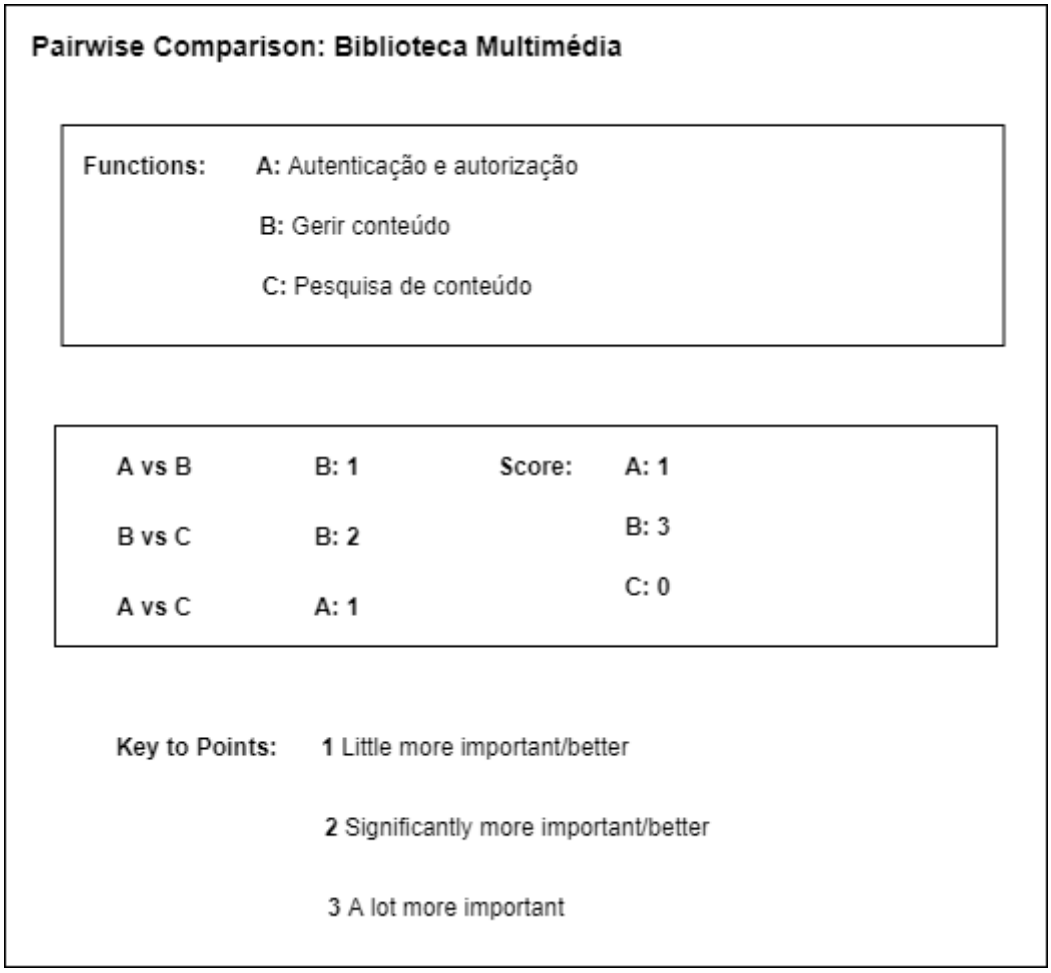



Figura 27 – *Pairwise Comparison*

Como se pode verificar na figura, o resultado indica que a função com maior importância e pontuação é a de Gerir conteúdo, com 3 pontos, seguido da função de Autenticação e autorização (1 ponto) e a de Pesquisa de conteúdo (0 pontos).

### 3.2.1 *Quality Function Deployment* (QFD)

O *Quality Function Deployment* é um "sistema para desenhar um produto ou serviço baseado nas exigências do cliente que envolve todos os membros do fabricante ou da organização fornecedora" (Warwick Manufacturing Group, 2007). Esta técnica será aplicada de forma a analisar os requisitos de qualidade mais relevantes para o cliente em conjunto com os requisitos técnicos do produto. Para isso, é apresentado na Figura 28 a Casa de Qualidade que demonstra os valores da relação entre os requisitos do cliente e as características técnicas do produto.

De forma a aplicar a Casa de Qualidade é necessário definir previamente os requisitos de qualidade do cliente ("*Whats*"):

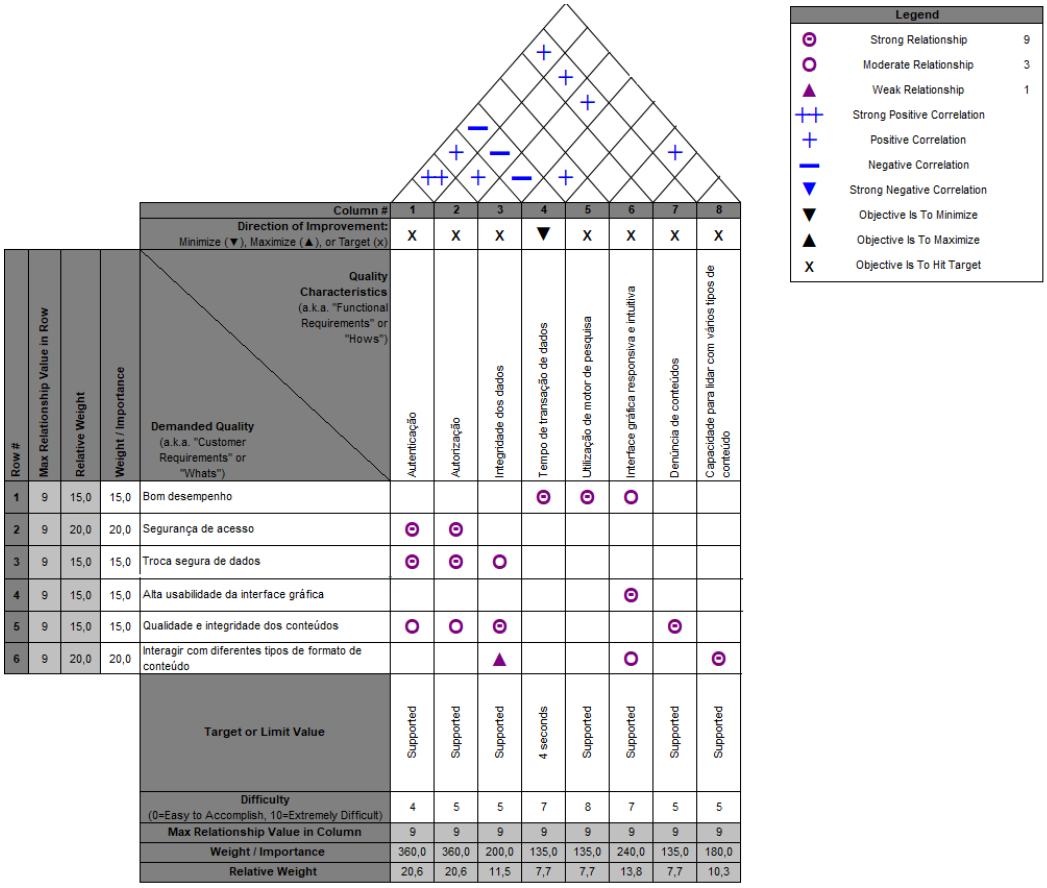
- Bom desempenho;

- Segurança de acesso;
- Troca segura de dados;
- Alta usabilidade da interface gráfica;
- Qualidade e integridade dos conteúdos;
- Rápida e eficaz capacidade de resposta na pesquisa de conteúdos;
- Interagir com diferentes tipos de formato de conteúdo.

Para além dos requisitos de qualidade do cliente é necessário também definir as características técnicas do produto ("*Hows*"):

- Autenticação;
- Autorização;
- Integridade dos dados;
- Tempo de transação de dados;
- Motor de pesquisa;
- Interface gráfica responsiva e intuitiva;
- Denúncia de conteúdos;
- Capacidade para lidar com vários tipos de conteúdo.

Estabelecidos os requisitos procedeu-se à elaboração da Casa de Qualidade, apresentada na Figura 28.



| Legend | | |
|---|---|---|
| ⊙ | Strong Relationship | 9 |
| ○ | Moderate Relationship | 3 |
| ▲ | Weak Relationship | 1 |
| ++ | Strong Positive Correlation | |
| + | Positive Correlation | |
| − | Negative Correlation | |
| ▼ | Strong Negative Correlation | |
| ▼ | Objective Is To Minimize | |
| ▲ | Objective Is To Maximize | |
| X | Objective Is To Hit Target | |

| Row # | Max Relationship Value in Row | Relative Weight | Weight / Importance | Quality Characteristics (a.k.a. "Functional Requirements" or "Hows") / Demanded Quality (a.k.a. "Customer Requirements" or "Whats") | 1 Autenticação | 2 Autorização | 3 Integridade dos dados | 4 Tempo de transação de dados | 5 Utilização de motor de pesquisa | 6 Interface gráfica responsiva e intuitiva | 7 Denúncia de conteúdos | 8 Capacidade para lidar com vários tipos de conteúdo |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | Direction of Improvement: Minimize (▼), Maximize (▲), or Target (x) | X | X | X | ▼ | X | X | X | X |
| 1 | 9 | 15,0 | 15,0 | Bom desempenho | | | | ⊙ | ⊙ | ○ | | |
| 2 | 9 | 20,0 | 20,0 | Segurança de acesso | ⊙ | ⊙ | | | | | | |
| 3 | 9 | 15,0 | 15,0 | Troca segura de dados | ⊙ | ⊙ | ○ | | | | | |
| 4 | 9 | 15,0 | 15,0 | Alta usabilidade da interface gráfica | | | | | | ⊙ | | |
| 5 | 9 | 15,0 | 15,0 | Qualidade e integridade dos conteúdos | ○ | ○ | ⊙ | | | | ⊙ | |
| 6 | 9 | 20,0 | 20,0 | Interagir com diferentes tipos de formato de conteúdo | | | ▲ | | | ○ | | ⊙ |
| | | | | Target or Limit Value | Supported | Supported | Supported | 4 seconds | Supported | Supported | Supported | Supported |
| | | | | Difficulty (0=Easy to Accomplish, 10=Extremely Difficult) | 4 | 5 | 5 | 7 | 8 | 7 | 5 | 5 |
| | | | | Max Relationship Value in Column | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 |
| | | | | Weight / Importance | 360,0 | 360,0 | 200,0 | 135,0 | 135,0 | 240,0 | 135,0 | 180,0 |
| | | | | Relative Weight | 20,6 | 20,6 | 11,5 | 7,7 | 7,7 | 13,8 | 7,7 | 10,3 |

Figura 28 – Casa de Qualidade

Relativamente à avaliação numérica atribuída no peso/importância dos requisitos de qualidade do cliente, os valores indicados têm origem num inquérito efetuado a colaboradores da empresa, tendo sido pedido para atribuírem um grau de importância a cada requisito proposto, considerando que a soma de todos os valores teria de ser 100. Assim, foi atribuído um valor superior (20) ao requisito de segurança de acesso e ao requisito de interagir com diferentes tipos de formatos de conteúdo. Para o primeiro, é importante garantir a restrição do acesso a certas funcionalidades da nova plataforma, permitindo que só utilizadores com um grau específico de autorização possam aceder a essas funcionalidades. Para o segundo, a ALERT contém conteúdos produzidos em mais do que um tipo de formato e um determinado tema pode ser constituído por mais do que um tipo de conteúdo, pelo que é importante que a plataforma seja capaz de funcionar com diferentes formatos, caso contrário o potencial desta aplicação *web* seria consideravelmente mais baixo por restringir a divulgação de conteúdos a um só formato específico, desvalorizando o conhecimento que pode ser transmitido com recurso a mais tipos de conteúdo. Com uma ligeira menor importância comparativamente com os anteriores, foi atribuído o mesmo peso aos restantes quatro requisitos (15) pois, uma vez que nenhum desses quatro requisitos foi considerado mais relevante, foi atribuído quantitativamente o mesmo grau de importância no que diz respeito aos requisitos de qualidade do cliente.

Analisando os resultados da aplicação da técnica QFD e da Casa de Qualidade na Figura 28, em termos de desempenho os aspetos técnicos mais importantes a considerar são o tempo de transição dos dados e a utilização de um motor de pesquisa, uma vez que a pesquisa e recolha de conteúdos será feita com recurso a esse motor. Para o produto ter um bom desempenho é também necessário que possua uma interface gráfica rápida e responsiva. Relativamente à segurança de acesso, esta pode ser imposta através da autenticação e autorização. No que diz respeito à troca segura de dados, a autenticação, autorização e integridade dos dados são as características principais para garantir que os dados não são alterados nem corrompidos durante uma transação. Para a interface gráfica ser de alta usabilidade, o produto deve aplicar no seu desenvolvimento uma interface gráfica responsiva e intuitiva. Sobre a qualidade e integridade dos conteúdos a adicionar ao produto, este requisito depende das características de autenticação, autorização, integridade dos dados e da denúncia de conteúdos. Com a autenticação e autorização é assumida uma qualidade relativa do conteúdo adicionado, uma vez que o utilizador tem de estar autenticado e ter autorização para submeter um conteúdo. Com a integridade dos dados garante-se que o conteúdo chega ao seu destino mantendo a sua informação e com a função de denúncia de um conteúdo assegura-se a sua qualidade. Por fim, é pedido que o cliente possa interagir com diferentes tipos de formatos de conteúdo, pelo que o produto, a nível de interface, deve adotar uma interface gráfica responsiva apta a apresentar diferentes tipos de conteúdo, e a nível interno do produto, este deve ter capacidade para lidar com vários tipos de conteúdo.

Sobre a relação entre os requisitos de qualidade do cliente e as características técnicas do produto, verifica-se que a autenticação tem uma forte relação com a autorização, uma vez que ambos os requisitos se complementam para aumentar a segurança do sistema. A autenticação relaciona-se também positivamente com a integridade dos dados e com a denúncia de conteúdos, dado que promove a segurança ao restringir o acesso dos utilizadores a esta funcionalidade. Tal como a autenticação, a autorização aumenta a segurança dos mesmos requisitos. Pelo lado negativo, observa-se que a autenticação e autorização podem aumentar o tempo total de transação de dados devido à necessidade de efetuar mais operações de condições de segurança. Sobre a relação entre outras características técnicas, o tempo de transação de dados tem uma relação positiva com a utilização do motor de pesquisa, uma vez que este permite reduzir o tempo de transação e a quantidade de dados movimentados. A interface gráfica responsiva e intuitiva é responsável por ter contacto direto com o utilizador e, portanto, é um dos requisitos com impacto na capacidade do sistema para lidar com vários tipos de conteúdo.

## 3.3 Criação de Alternativas

Nesta etapa são definidas as possíveis alternativas existentes para o desenvolvimento das funcionalidades do produto. No contexto deste projeto, foram consideradas algumas alternativas de solução principalmente para aquela que é uma das principais funções do

produto para o cliente, a função de pesquisa de conteúdo. Assim, foram selecionadas duas ferramentas externas para adicionar a capacidade de pesquisa ao produto:

- Solr;
- Elasticsearch.

Estas ferramentas e as suas vertentes técnicas encontram-se descritas na secção 2.3.

## 3.4 Análise e Avaliação

De forma a analisar e avaliar as alternativas atrás identificadas, foi aplicado o Método de Análise Hierárquica (AHP). Este método, criado pelo professor Thomas L. Saaty em 1980, permite o uso de critérios qualitativos bem como quantitativos no processo de avaliação, dividindo o problema de decisão em níveis hierárquicos facilitando, assim, a sua compreensão e avaliação (Nicola, 2018). Para comparar as alternativas (Solr e Elasticsearch) foram usados os seguintes critérios:

- Suporte;
- Facilidade de configuração e implementação;
- Funções de pesquisa;
- Velocidade de pesquisa.

O critério de Suporte centraliza-se no apoio da comunidade e documentação existente sobre os aspetos técnicos das alternativas. Este critério é importante pois significa mais ou menos tempo despendido em ultrapassar as dificuldades encontradas. O critério de Facilidade de configuração e implementação traduz-se, tal como o nome indica, na facilidade com que se configura a ferramenta para iniciar o seu uso e se implementa os vários aspetos das funções de pesquisa. O critério Funções de pesquisa refere-se às capacidades e diferentes serviços disponibilizados pelas alternativas na pesquisa de conteúdos como, por exemplo, capacidades de filtração dos dados pesquisados. Por último, o critério Velocidade de pesquisa aborda a velocidade/tempo com que são obtidos os resultados da pesquisa efetuada. Para as alternativas consideradas, este critério poderia ser dividido em duas partes, velocidade de pesquisa de dados estáticos e velocidade de pesquisa de dados dinâmicos, mas no contexto deste projeto o critério foi interpretado de uma forma geral, atribuindo um pouco mais de peso à componente da velocidade com dados estáticos nos valores de comparação atribuídos.

### 3.4.1 Método de Análise Hierárquica (AHP)

Procedendo à aplicação do AHP será então analisado qual a alternativa mais adequada a usar de acordo com os critérios e valores de comparação estabelecidos. Para aplicar este método deve-se primeiramente construir a árvore hierárquica de decisão, onde é indicado o objetivo, os critérios e as alternativas sobre a forma de hierarquia (Nicola, 2018). A árvore construída é apresentada na Figura 29.

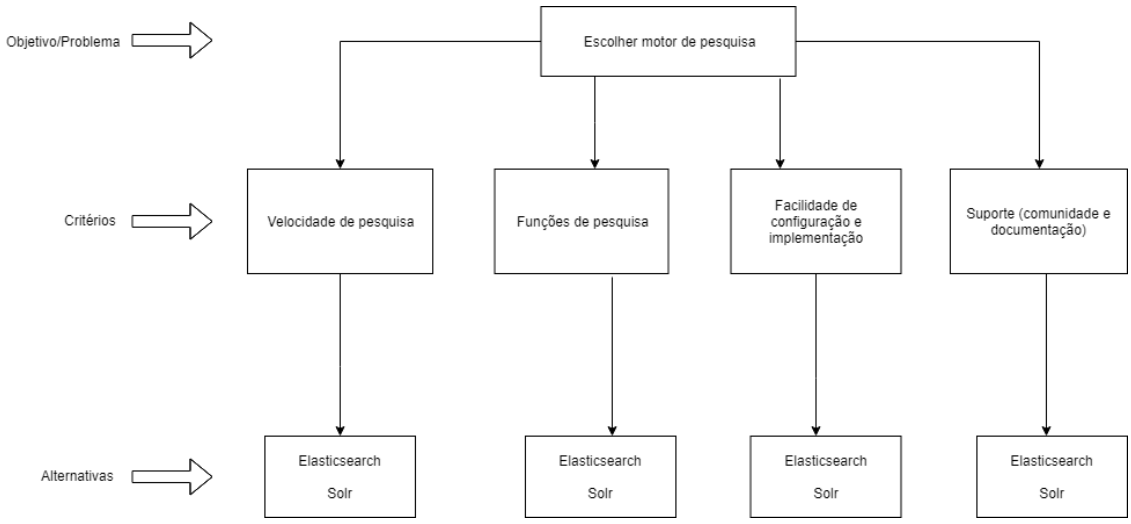



Figura 29 – Árvore hierárquica de decisão

A fase seguinte, de comparação de alternativas e critérios, consiste em estabelecer prioridades entre os elementos para cada nível da hierarquia através de uma matriz de comparação (Nicola, 2018). Nesta fase, para atribuir valores de comparação entre critérios foi usada a Escala Fundamental (Saaty, 2008) definida por Saaty. Esta escala, apresentada na Tabela 4, representa os níveis de importância de comparações.

Tabela 4 – Escala Fundamental (Nicola, 2018)

| Nível de importância | Definição | Explicação |
|---|---|---|
| 1 | Igual importância | As duas atividades contribuem igualmente para o objetivo |
| 3 | Fraca importância | A experiência e o julgamento favorecem levemente uma atividade em relação à outra |
| 5 | Forte importância | A experiência e o julgamento favorecem fortemente uma atividade em relação à outra |
| 7 | Muito forte importância | Uma atividade é fortemente favorecida em relação a outra |

| Nível de importância | Definição | Explicação |
|---|---|---|
| 9 | Importância absoluta | A evidência favorece uma atividade em relação a outra com o mais alto grau de certeza |
| 2,4,6,8 | Valores intermediários | Quando se procura uma condição de compromisso entre duas definições |

Na Tabela 5 são então atribuídos os valores de comparação de critérios.

Tabela 5 – Comparação de critérios

| | Suporte | Facilidade de configuração e implementação | Funções de pesquisa | Velocidade de pesquisa |
|---|---|---|---|---|
| Suporte | 1 | 2 | $\frac{1}{3}$ | $\frac{1}{2}$ |
| Facilidade de configuração e implementação | $\frac{1}{2}$ | 1 | $\frac{1}{4}$ | $\frac{1}{3}$ |
| Funções de pesquisa | 3 | 4 | 1 | 2 |
| Velocidade de pesquisa | 2 | 3 | $\frac{1}{2}$ | 1 |
| Soma | $\frac{13}{2}$ | 10 | $\frac{25}{12}$ | $\frac{23}{6}$ |

Como se pode verificar, foi considerado que o critério de Funções de pesquisa tem maior importância sobre todos os restantes critérios. A partir destes valores e com o objetivo de igualar todos os critérios a uma mesma unidade, procedeu-se à normalização desses valores, calculada através da divisão do valor atribuído a cada critério pela soma total da sua respetiva coluna. Na Tabela 6 são apresentados os valores normalizados.

Tabela 6 - Comparação de critérios com valores normalizados

| | Suporte | Facilidade de configuração e implementação | Funções de pesquisa | Velocidade de pesquisa |
|---|---|---|---|---|
| Suporte | $\frac{2}{13}$ | $\frac{1}{5}$ | $\frac{4}{25}$ | $\frac{3}{23}$ |

| | Suporte | Facilidade de configuração e implementação | Funções de pesquisa | Velocidade de pesquisa |
|---|---|---|---|---|
| Facilidade de configuração e implementação | $\frac{1}{13}$ | $\frac{1}{10}$ | $\frac{3}{25}$ | $\frac{2}{23}$ |
| Funções de pesquisa | $\frac{6}{13}$ | $\frac{2}{5}$ | $\frac{12}{25}$ | $\frac{12}{23}$ |
| Velocidade de pesquisa | $\frac{4}{13}$ | $\frac{3}{10}$ | $\frac{6}{25}$ | $\frac{6}{23}$ |

Normalizados todos os valores, é calculada a prioridade relativa para cada critério de forma a obter o vetor de prioridades. Este passo tem como objetivo identificar a ordem de importância de cada critério, calculando a média aritmética dos valores de cada linha normalizada no passo anterior. Na Tabela 7 é apresentada a prioridade relativa dos critérios.

Tabela 7 - Prioridade relativa dos critérios

| | Suporte | Facilidade de configuração e implementação | Funções de pesquisa | Velocidade de pesquisa | Prioridade Relativa |
|---|---|---|---|---|---|
| Suporte | $\frac{2}{13}$ | $\frac{1}{5}$ | $\frac{4}{25}$ | $\frac{3}{23}$ | 0,161 |
| Facilidade de configuração e implementação | $\frac{1}{13}$ | $\frac{1}{10}$ | $\frac{3}{25}$ | $\frac{2}{23}$ | 0,096 |
| Funções de pesquisa | $\frac{6}{13}$ | $\frac{2}{5}$ | $\frac{12}{25}$ | $\frac{12}{23}$ | 0,466 |
| Velocidade de pesquisa | $\frac{4}{13}$ | $\frac{3}{10}$ | $\frac{6}{25}$ | $\frac{6}{23}$ | 0,277 |

Através dos resultados obtidos do peso dos critérios é possível verificar que o critério que aparece em primeiro lugar é as Funções de pesquisa (46,6%), seguido da Velocidade da pesquisa (27,7%), Suporte (16,1%) e Facilidade de configuração e implementação do motor de pesquisa (9,6%).

Depois de obter os valores de prioridade relativa a etapa seguinte consiste em calcular a Razão de Consistência (RC) para medir quanto os julgamentos foram consistentes em relação a grandes amostras de juízos completamente aleatórios. A fórmula para calcular a RC é $RC = \frac{IC}{IR}$

, onde IC é o Índice de consistência e é obtido através da fórmula $IC = \frac{\lambda \max - n}{n-1}$. O n é a ordem da matriz e $\lambda \max$ é o maior valor próprio da matriz de comparação par a par. O IR (Índice Aleatório) corresponde a um índice aleatório calculado para matrizes quadradas de ordem n pelo Laboratório Nacional de Oak Ridge, nos EUA. Na Tabela 8 encontram-se definidos os valores de IR em função do número de critérios.

Tabela 8 – Valores de IR para matrizes quadradas de ordem n (Laboratório Nacional de Oak Ridge)

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0.00 | 0.00 | 0.58 | 0.90 | 1.12 | 1.24 | 1.32 | 1.41 | 1.45 | 1.49 | 1.51 | 1.48 | 1.56 | 1.57 | 1.59 |

Para calcular a RC é necessário em primeiro lugar obter o valor de $\boldsymbol{\lambda max}$, a partir da equação $\boldsymbol{Ax = \lambda max . x}$, sendo *x* o vetor de prioridades.

$$\begin{vmatrix} 1 & 2 & 0{,}333 & 0{,}5 \\ 0{,}5 & 1 & 0{,}250 & 0{,}333 \\ 3 & 4 & 1 & 2 \\ 2 & 3 & 0{,}5 & 1 \end{vmatrix} \times \begin{vmatrix} 0{,}161 \\ 0{,}096 \\ 0{,}466 \\ 0{,}277 \end{vmatrix} \cong \boldsymbol{\lambda max} \begin{vmatrix} 0{,}161 \\ 0{,}096 \\ 0{,}466 \\ 0{,}277 \end{vmatrix} \quad (1)$$

$$\begin{vmatrix} 0{,}647 \\ 0{,}385 \\ 1{,}887 \\ 1{,}120 \end{vmatrix} \cong \boldsymbol{\lambda max} \begin{vmatrix} 0{,}161 \\ 0{,}096 \\ 0{,}466 \\ 0{,}277 \end{vmatrix} \quad (2)$$

$$\boldsymbol{\lambda max} = 4{,}030 \quad (3)$$

Calculado o $\lambda max$ deve-se proceder ao cálculo do IC.

$$IC = \frac{4{,}030 - 4}{4 - 1} = 0{,}010$$

Com o valor do IC obtém-se o valor da RC. Se este valor for superior a 0,1 considera-se que os julgamentos não são confiáveis porque estão demasiado perto para o conforto de aleatoriedade e, neste caso, os resultados obtidos não apresentam valores consistentes.

$$RC = \frac{0{,}010}{0{,}90} = 0{,}011$$

Dado que 0,011 < 0,1 pode-se concluir que os valores das prioridades relativas são consistentes.

A fase seguinte do AHP consiste em obter o vetor de prioridades para cada critério, considerando cada uma das alternativas existentes e repetindo o processo de normalização e obtenção das prioridades relativas.

Primeiramente, será obtido o vetor de prioridades para o critério de Suporte. Na Tabela 9 são apresentados os valores de comparação de alternativas para este critério.

Tabela 9 - Comparação de alternativas para o critério de Suporte

| Suporte | Solr | Elasticsearch |
|---|---|---|
| Solr | 1 | 2 |
| Elasticsearch | $\frac{1}{2}$ | 1 |
| Soma | $\frac{3}{2}$ | 3 |

Na Tabela 10 são apresentados os valores atrás definidos normalizados.

Tabela 10 - Comparação de alternativas para o critério de Suporte com valores normalizados

| Suporte | Solr | Elasticsearch |
|---|---|---|
| Solr | $\frac{2}{3}$ | $\frac{2}{3}$ |
| Elasticsearch | $\frac{1}{3}$ | $\frac{1}{3}$ |

Com os valores normalizados, é então calculada a prioridade relativa das alternativas para o critério de Suporte (Tabela 11).

Tabela 11 - Prioridade Relativa das alternativas para o critério de Suporte

| Suporte | Solr | Elasticsearch | Prioridade Relativa |
|---|---|---|---|
| Solr | $\frac{2}{3}$ | $\frac{2}{3}$ | 0,666 |
| Elasticsearch | $\frac{1}{3}$ | $\frac{1}{3}$ | 0,334 |

Para as alternativas e o critério de Suporte obteve-se, então, o vetor próprio $\begin{vmatrix}0{,}666\\0{,}334\end{vmatrix}$.

De seguida efetua-se o mesmo processo para o critério de Facilidades de configuração e implementação. Na Tabela 12 são apresentados os valores de comparação de alternativas para este critério.

Tabela 12 - Comparação de alternativas para o critério de Facilidade de configuração e implementação

| Facilidade de configuração e implementação | Solr | Elasticsearch |
|---|---|---|
| Solr | 1 | $\frac{1}{3}$ |

| Facilidade de configuração e implementação | Solr | Elasticsearch |
|---|---|---|
| Elasticsearch | 3 | 1 |
| Soma | 4 | $\frac{4}{3}$ |

Na Tabela 13 são apresentados os valores de comparação de alternativas normalizados.

Tabela 13 - Comparação de alternativas para o critério de Facilidade de configuração e implementação com valores normalizados

| Facilidade de configuração e implementação | Solr | Elasticsearch |
|---|---|---|
| Solr | $\frac{1}{4}$ | $\frac{1}{4}$ |
| Elasticsearch | $\frac{3}{4}$ | $\frac{3}{4}$ |

Após a normalização dos valores é obtida a prioridade relativa das alternativas (Tabela 14).

Tabela 14 - Prioridade Relativa das alternativas para o critério de Facilidade de configuração e implementação

| Facilidade de configuração e implementação | Solr | Elasticsearch | Prioridade Relativa |
|---|---|---|---|
| Solr | $\frac{1}{4}$ | $\frac{1}{4}$ | 0,250 |
| Elasticsearch | $\frac{3}{4}$ | $\frac{3}{4}$ | 0,750 |

O vetor próprio obtido das alternativas para o critério de Facilidade de configuração e implementação é $\begin{vmatrix} 0{,}250 \\ 0{,}720 \end{vmatrix}$.

O critério seguinte a obter o vetor de prioridades é o critério de Funções de pesquisa. Na Tabela 15 são apresentados os valores de comparação de alternativas para este critério.

Tabela 15 - Comparação de alternativas para o critério Funções de pesquisa

| Funções de pesquisa | Solr | Elasticsearch |
|---|---|---|
| Solr | 1 | 2 |

| Funções de pesquisa | Solr | Elasticsearch |
|---|---|---|
| Elasticsearch | $\frac{1}{2}$ | 1 |
| Soma | $\frac{3}{2}$ | 3 |

Na Tabela 16 são apresentados os valores de comparação de alternativas normalizados.

Tabela 16 - Comparação de alternativas para o critério Funções de pesquisa com valores normalizados

| Funções de pesquisa | Solr | Elasticsearch |
|---|---|---|
| Solr | $\frac{2}{3}$ | $\frac{2}{3}$ |
| Elasticsearch | $\frac{1}{3}$ | $\frac{1}{3}$ |

Na tabela seguinte (Tabela 17) é apresentada a prioridade relativa das alternativas para o critério Funções de pesquisa.

Tabela 17 - Prioridade Relativa das alternativas para o critério Funções de pesquisa

| Funções de pesquisa | Solr | Elasticsearch | Prioridade Relativa |
|---|---|---|---|
| Solr | $\frac{2}{3}$ | $\frac{2}{3}$ | 0,666 |
| Elasticsearch | $\frac{1}{3}$ | $\frac{1}{3}$ | 0,334 |

Através dos cálculos obtém-se o vetor de prioridade $\begin{vmatrix} 0{,}666 \\ 0{,}334 \end{vmatrix}$.

Por fim, procedeu-se com o mesmo processo para o último critério, a Velocidade de pesquisa. Na Tabela 18 são apresentados os valores de comparação de alternativas para este critério.

Tabela 18 – Comparação de alternativas para o critério Velocidade de pesquisa

| Velocidade de pesquisa | Solr | Elasticsearch |
|---|---|---|
| Solr | 1 | 1 |
| Elasticsearch | 1 | 1 |
| Soma | 2 | 2 |

Na tabela 19 são apresentados os valores atrás referidos normalizados.

Tabela 19 - Comparação de alternativas para o critério Velocidade de Pesquisa com valores normalizados

| Velocidade de pesquisa | Solr | Elasticsearch |
|---|---|---|
| Solr | $\frac{1}{2}$ | $\frac{1}{2}$ |
| Elasticsearch | $\frac{1}{2}$ | $\frac{1}{2}$ |

Na Tabela 20 pode-se visualizar a prioridade relativa das alternativas para o critério Velocidade de pesquisa.

Tabela 20 - Prioridade relativa de alternativas para o critério Velocidade de pesquisa

| Velocidade de pesquisa | Solr | Elasticsearch | Prioridade Relativa |
|---|---|---|---|
| Solr | $\frac{1}{2}$ | $\frac{1}{2}$ | 0,500 |
| Elasticsearch | $\frac{1}{2}$ | $\frac{1}{2}$ | 0,500 |

Assim, o vetor de prioridades das alternativas para o critério Velocidade de pesquisa é $\begin{vmatrix}0{,}500\\0{,}500\end{vmatrix}$.

Depois de calculado o vetor de prioridades das alternativas estão reunidos todos os valores necessários para proceder ao cálculo final para determinar qual o motor de pesquisa mais adequado de acordo com o peso dos critérios e os valores dos vetores de prioridades. Os valores obtidos anteriormente podem ser visualizados na Figura 30.

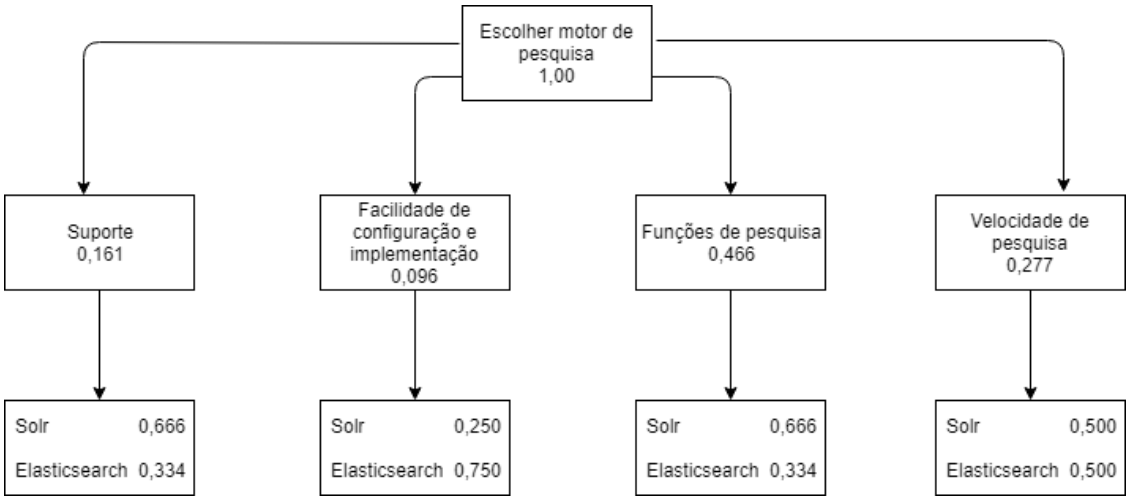



Figura 30 – Motores de pesquisa e critérios

O cálculo final consiste em obter a prioridade composta para as alternativas:

$$\begin{vmatrix} 0{,}666 & 0{,}250 & 0{,}666 & 0{,}500 \\ 0{,}334 & 0{,}750 & 0{,}334 & 0{,}500 \end{vmatrix} \times \begin{vmatrix} 0{,}161 \\ 0{,}096 \\ 0{,}466 \\ 0{,}277 \end{vmatrix} = \begin{vmatrix} 0{,}580 \\ 0{,}420 \end{vmatrix}$$

Observando os resultados obtidos é possível concluir que a escolha mais adequada incide sobre o motor de pesquisa Solr, visto que obteve o valor mais elevado (0,580) comparativamente com a outra alternativa (0,420). Para além disso, o Solr é usado como motor de pesquisa noutros programas de *software* da empresa, pelo que a sua aplicação neste projeto é facilitada pelo conhecimento já existente dos colaboradores na manipulação desta ferramenta. Analisando com maior detalhe os resultados obtidos, o Solr supera o Elasticsearch nos critérios de suporte e de funções de pesquisa, enquanto o Elasticsearch revela-se superior quanto à sua facilidade de configuração e implementação. Relativamente à velocidade de pesquisa, verifica-se um empate entre os dois sistemas. Excluindo este critério em que se verifica um empate e dado que o Solr apresenta um domínio nos critérios com maior peso na avaliação, observa-se então um favorecimento do Solr sobre o Elasticsearch.

## 3.5 Implementação

A última fase do processo de análise de valor consiste na implementação do produto. No contexto deste projeto, esta fase traduz-se na implementação das funcionalidades do produto identificadas, tendo em conta os resultados obtidos na análise AHP efetuada anteriormente.

# 4 Análise

Neste capítulo é apresentada a fase de análise do projeto, mais concretamente os requisitos a ter em conta no desenvolvimento da plataforma, bem como o seu modelo de domínio e o seu modelo de dados. Para além disso, são também apresentadas maquetes que foram desenhadas como base para a interface gráfica a implementar.

## 4.1 Requisitos

Em qualquer projeto de *software* é sempre importante analisar e pensar nos requisitos para conhecer e satisfazer as necessidades do cliente, bem como visualizar as características técnicas e de qualidade necessárias a ter em conta quanto à solução a desenvolver. Assim, procedeu-se à análise dos requisitos funcionais e não funcionais, usando como base as categorias referenciadas no modelo FURPS+ que se enquadram no contexto deste projeto. Este modelo é usado para classificar atributos de qualidade de *software* (Eeles, 2005).

### 4.1.1 Funcionais

Considerando a análise feita para este projeto e os interesses da ALERT em relação ao desenvolvimento da Biblioteca Multimédia, foram definidas funcionalidades e respetivos requisitos sobre a forma de casos de uso. Estes são apresentados através de duas vistas diferentes, na Tabela 21 e no diagrama de casos de uso da Figura 31. Relativamente aos atores do sistema, existem quatro atores diferentes: *Admin, High User* e *User,* que são utilizadores registados e autenticados na plataforma, e o utilizador não autenticado, sendo que a principal diferença entre estes são as permissões de acesso a certas funcionalidades do sistema, visíveis na tabela seguinte.

Tabela 21 – Requisitos funcionais do sistema

| Funcionalidade | Caso de Uso | Admin | High User | User | Utilizador não autenticado | Sistema |
|---|---|---|---|---|---|---|
| Gerir conteúdo | Visualizar conteúdos | Sim | Sim | Sim | Sim | - |
| | Visualizar detalhes do conteúdo | Sim | Sim | Sim | Sim | - |
| | Contabilizar número de visualizações do conteúdo | - | - | - | - | Sim |
| | Criar conteúdo | Sim | Sim | Não | Não | - |
| | Editar conteúdo | Sim[3] | Sim[4] | Não | Não | - |
| | Eliminar conteúdo | Sim[3] | Sim[4] | Não | Não | - |
| | Partilhar conteúdo por email | Sim | Sim | Sim | Sim | - |
| | Partilhar conteúdo por WhatsApp | Sim | Sim | Sim | Sim | - |
| | Classificar conteúdo | Sim | Sim | Sim | Não | - |
| Gerir conteúdos denunciados | Denunciar conteúdo | Sim | Sim | Sim | Não | - |
| | Visualizar conteúdo denunciado | Sim[5] | Sim[5] | Não | Não | - |
| | Adicionar mensagem ao registo da denúncia | Sim[5] | Sim[5] | Não | Não | - |
| | Eliminar denúncia | Sim[5] | Sim[5] | Não | Não | - |
| Autenticação | Efetuar autenticação | Sim | Sim | Sim | Sim | - |
| Gerir Favoritos | Visualizar Favoritos | Sim | Sim | Sim | Não | - |
| | Adicionar conteúdo aos seus Favoritos | Sim | Sim | Sim | Não | - |
| | Remover conteúdo dos seus Favoritos | Sim | Sim | Sim | Não | - |
| Gerir Histórico | Consultar o seu histórico de visualização de conteúdos | Sim | Sim | Sim | Não | - |
| | Remover conteúdo do seu histórico de visualização | Sim | Sim | Sim | Não | - |
| | Adicionar conteúdo visualizado ao histórico do utilizador | - | - | - | - | Sim |
| Gerir *Playlist* | Visualizar *Playlist* | Sim | Sim | Sim | Sim | - |
| | Criar *Playlist* | Sim | Sim | Sim | Não | - |
| | Editar *Playlist* | Sim[3] | Sim[4] | Sim | Não | - |

[3] De todos os utilizadores
[4] Apenas criado por si
[5] Se reportado para si

| Funcionalidade | Caso de Uso | Admin | High User | User | Utilizador não autenticado | Sistema |
|---|---|---|---|---|---|---|
| | Eliminar *Playlist* | Sim[3] | Sim[4] | Sim | Não | - |
| | Adicionar conteúdo à *Playlist* | Sim | Sim | Sim | Não | - |
| | Remover conteúdo da *Playlist* | Sim | Sim | Sim | Não | - |
| | Partilhar *Playlist* por email | Sim | Sim | Sim | Sim | - |
| | Partilhar *Playlist* por WhatsApp | Sim | Sim | Sim | Sim | - |
| | Partilhar cópia da *Playlist* com outros utilizadores | Sim | Sim | Sim | Não | - |
| | Aceitar/Rejeitar cópia da *Playlist* partilhada | Sim | Sim | Sim | Não | - |
| Perfil de Utilizador | Visualizar perfil de utilizador registado | Sim | Sim | Sim | Sim | - |
| | Visualizar conteúdos publicados por utilizador registado | Sim | Sim | Sim | Sim | - |
| | Visualizar Favoritos de utilizador registado | Sim | Sim | Sim | Sim | - |
| | Visualizar *Playlists* de utilizador registado | Sim | Sim | Sim | Sim | - |
| | Contactar utilizador por email | Sim | Sim | Sim | Sim | - |
| Pesquisa | Pesquisar por conteúdo | Sim | Sim | Sim | Sim | - |
| | Filtrar resultados da pesquisa | Sim | Sim | Sim | Sim | - |

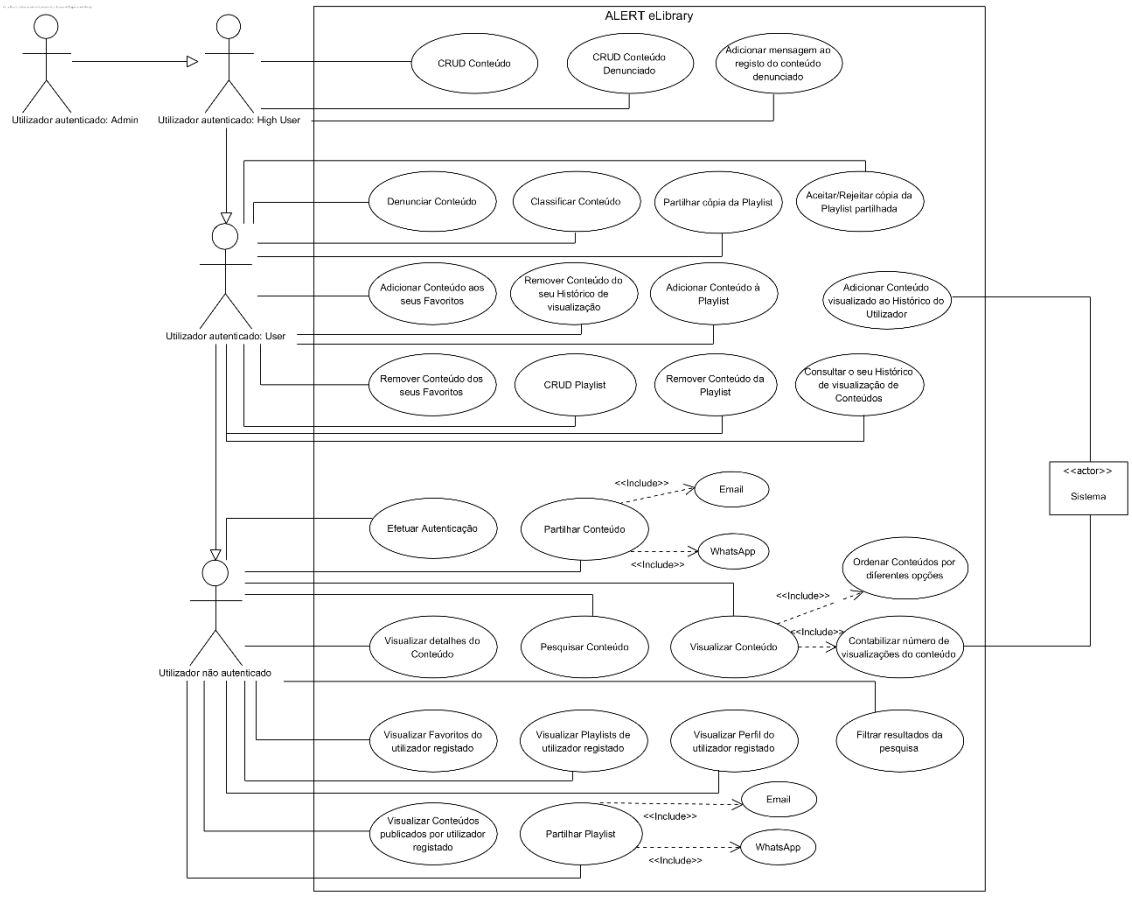



Figura 31 – Diagrama de casos de uso

De seguida é efetuada uma breve descrição das funcionalidades pretendidas:

- Gerir conteúdo: deve ser possível visualizar os conteúdos existentes navegando pelas diferentes áreas e categorias de conteúdos definidas e ordenando os conteúdos apresentados segundo diferentes opções (data, número de visualizações e classificação do conteúdo). Ao selecionar um conteúdo, o utilizador tem acesso a uma vista com mais detalhes dos metadados e estatísticas do conteúdo consultado. Os utilizadores autenticados no sistema e com permissões para tal, podem efetuar a adição dos seus próprios conteúdos na plataforma e geri-los (editar e eliminar). Cada vez que um conteúdo é visualizado, o sistema deve incrementar o número de visualizações desse conteúdo. Na página de detalhes do conteúdo, o sistema deve permitir a atribuição de uma classificação ao conteúdo (*rating*) e qualquer utilizador da plataforma pode optar por enviar a ligação para o conteúdo por email ou por WhatsApp.
- Gerir conteúdos denunciados: um utilizador, autenticado no sistema, ao visualizar um conteúdo na sua vista de detalhes, deve ter a possibilidade de denunciar um conteúdo, registando uma denúncia indicando o assunto e uma mensagem e selecionando o destinatário da denúncia, podendo esse ser o utilizador que publicou o conteúdo e/ou os administradores. O recetor da denúncia deve poder consultar todas as informações

da denúncia efetuada e do conteúdo alvo, tendo como hipótese eliminar a denúncia e/ou o conteúdo, bem como editá-lo. Adicionalmente, deve existir uma opção de adicionar uma mensagem à denúncia.

- Autenticação: os utilizadores registados no sistema podem efetuar autenticação através do seu *username* e *password*.
- Gerir favoritos: os utilizadores autenticados devem ter a possibilidade de adicionar e remover um conteúdo à sua lista de favoritos na vista de detalhes do conteúdo, bem como terem uma área de visualização e gestão dos seus conteúdos favoritos.
- Gerir histórico: um utilizador autenticado no sistema deve ter uma área de consulta dos conteúdos que visualizou enquanto autenticado no sistema, podendo eliminar o seu histórico quando desejar. Ao visualizar um conteúdo, o sistema deve adicionar esse conteúdo ao histórico de visualização do utilizador.
- Gerir *Playlist*: um utilizador autenticado deve ter uma área dedicada para a visualização e gestão das suas *Playlists*. Para cada *Playlist*, o utilizador pode visualizar e navegar pelos conteúdos associados, bem como adicionar mais conteúdos ou removê-los da *Playlist*. Um utilizador autenticado pode enviar uma cópia da *Playlist* selecionada para outros utilizadores registados no sistema, bem como partilhar a ligação para a visualização dessa *Playlist* por email ou WhatsApp. O utilizador cuja cópia foi enviada, deve visualizar na sua área de mensagens o pedido de receção dessa cópia, podendo aceitar ou rejeitar esse pedido. Ao aceitar a cópia, o utilizador deve passar a ter essa *Playlist* disponível nas suas *Playlists*.
- Perfil de Utilizador: deve ser possível aceder à área do próprio perfil de utilizador ou de outro utilizador, onde devem ser mostrados os últimos conteúdos por ele publicados, os seus favoritos e *Playlists* mais recentes, bem como alguns dados (nome, país, contactar por email) e estatísticas do utilizador (número de conteúdos publicados, número de visualizações dos conteúdos, número de conteúdos favoritos e número de classificações atribuídas). Adicionalmente, para além de ver os conteúdos, favoritos e *Playlists* mais recentes, devem existir opções para visualizar todos os conteúdos, favoritos e *Playlists* do utilizador.
- Pesquisa: deve existir uma funcionalidade avançada de pesquisa de conteúdos, onde o utilizador pode pesquisar através de um campo de texto e refinar a sua pesquisa através da seleção de diferentes filtros (área, categoria e língua do conteúdo).

#### 4.1.1.1 Segurança

Relativamente à segurança, categoria inserida no "F" do modelo FURPS+, deve existir um processo de autenticação e verificação da autorização de acesso dos utilizadores a certas funcionalidades e informações. Deve ser dada atenção à submissão de ficheiros correspondentes aos conteúdos: deve ser definido um tamanho máximo para os ficheiros; o nome dos ficheiros deve ser modificado antes do seu armazenamento; os ficheiros submetidos devem ser armazenados numa localização segura desconhecida para o exterior. Relativamente ao protocolo de comunicação, caso a ALERT pretenda obter um certificado digital SSL (*Secure*

*Socket Layer*), deve ser usado o protocolo HTTPS para aumentar a segurança das transições de dados efetuadas.

### 4.1.2 Não Funcionais

Para o levantamento de requisitos não funcionais, como referido anteriormente, aplicaram-se as categorias referenciadas no modelo FURPS+ que se enquadram no contexto deste projeto.

#### 4.1.2.1 Usabilidade

O sistema deve apresentar um elevado nível de usabilidade de forma a permitir que os utilizadores executem as tarefas desejadas de forma fácil e rápida, proporcionando um maior nível de eficiência na execução das atividades. Uma plataforma amigável implica a redução de custos e tempo em eventuais formações de suporte ao seu uso, pelo que o sistema deve ser fácil de aprender, estando estruturado de uma forma que permita a navegação dos utilizadores intuitivamente pela lógica do seu conteúdo e funções. A sua interface deve ser intuitiva, responsiva e apelativa, deve permitir uma fácil memorização para a repetição de atividades de forma rápida e deve ser tolerante a erros, possibilitando a fácil correção dos mesmos.

#### 4.1.2.2 Confiabilidade

O sistema deve ser desenhado de forma a ser robusto, confiável e tolerante a erros, funcionando de modo fiável com o mínimo de interrupções possíveis após o seu lançamento. Deve garantir que os dados transacionados não são corrompidos e a sua integridade é mantida. A ferramenta de motor de pesquisa utilizada deve ter capacidade para correr mais do que uma instância se necessário, processando as respostas às pesquisas efetuadas de forma mais rápida, bem como assegurando o funcionamento se alguma das instâncias falharem (redundância).

#### 4.1.2.3 Desempenho

O motor de pesquisa utilizado deve ter capacidade para correr mais do que uma instância, armazenando dados em múltiplos nós e fornecendo resultados de pesquisa mais rapidamente. Na navegação do *website* da nova plataforma, o tempo de resposta das operações mais pesadas não deve ser muito demorado, caso contrário e, se possível, a operação deverá ser assíncrona. Considerando a hipótese de no futuro aumentar consideralvemente o número de utilizadores deste *software*, o sistema deve ser desenvolvido numa tecnologia fácil de escalar.

Relativamente ao consumo de recursos, o limite destes valores está dependente dos equipamentos utilizados pela ALERT no alojamento do *software*. Uma vez que serão guardados ficheiros submetidos pelo utilizador, é necessário ter em atenção o espaço disponível para ter capacidade de armazenamento suficiente.

#### 4.1.2.4 Suportabilidade

O sistema deve estar desenhado de acordo com bons princípios e padrões de *software* de forma a ser adaptado e facilmente extensível, possibilitando a adição de novas funcionalidades e serviços sem afetar os já existentes. Para além disso, o uso desses padrões e de boas práticas, bem como a elaboração de documentação técnica atualizada e comentários ao código, devem facilitar e reduzir a necessidade de manutenção do sistema. A interface gráfica e os serviços do sistema devem ser compatíveis com dispositivos móveis e com os principais navegadores de internet. A solução pensada deve possibilitar a autenticação dos utilizadores existentes no *Active Directory* (AD) da ALERT no novo sistema. O sistema deve integrar e ser compatível com o motor de pesquisa usado, comunicando com este para troca de informações. Deve ser possível efetuar diversos conjuntos de testes ao sistema para avaliar e garantir o seu correto funcionamento. O sistema deve ser implantado com facilidade com o auxílio de tecnologias existentes para esse propósito e deve poder ser acedido remotamente a partir de qualquer dispositivo com um navegador e ligação à internet.

#### 4.1.2.5 Restrições de Desenho

O desenho da solução para o problema tem algumas restrições no que consta às ferramentas de desenvolvimento aplicadas, uma vez que estas devem ir de encontro às ferramentas usadas no processo de desenvolvimento de *software* da ALERT. Assim, é necessário ter em conta no desenho da solução que o desenvolvimento da plataforma no lado do servidor está restringido à utilização da linguagem Java e a base de dados Oracle. No desenho da solução devem ser adotados bons princípios e padrões de engenharia de *software*.

#### 4.1.2.6 Requisitos de Implementação

A solução a implementar deve ser desenvolvida em linguagem Java (à exceção do lado cliente) e usar base de dados Oracle, seguindo assim as tecnologias e ferramentas de desenvolvimento de *software* adotadas na ALERT.

## 4.2 Modelo de Domínio

O modelo de domínio apresentado na Figura 32 representa as entidades de negócio necessárias para o desenvolvimento do sistema e as suas relações.

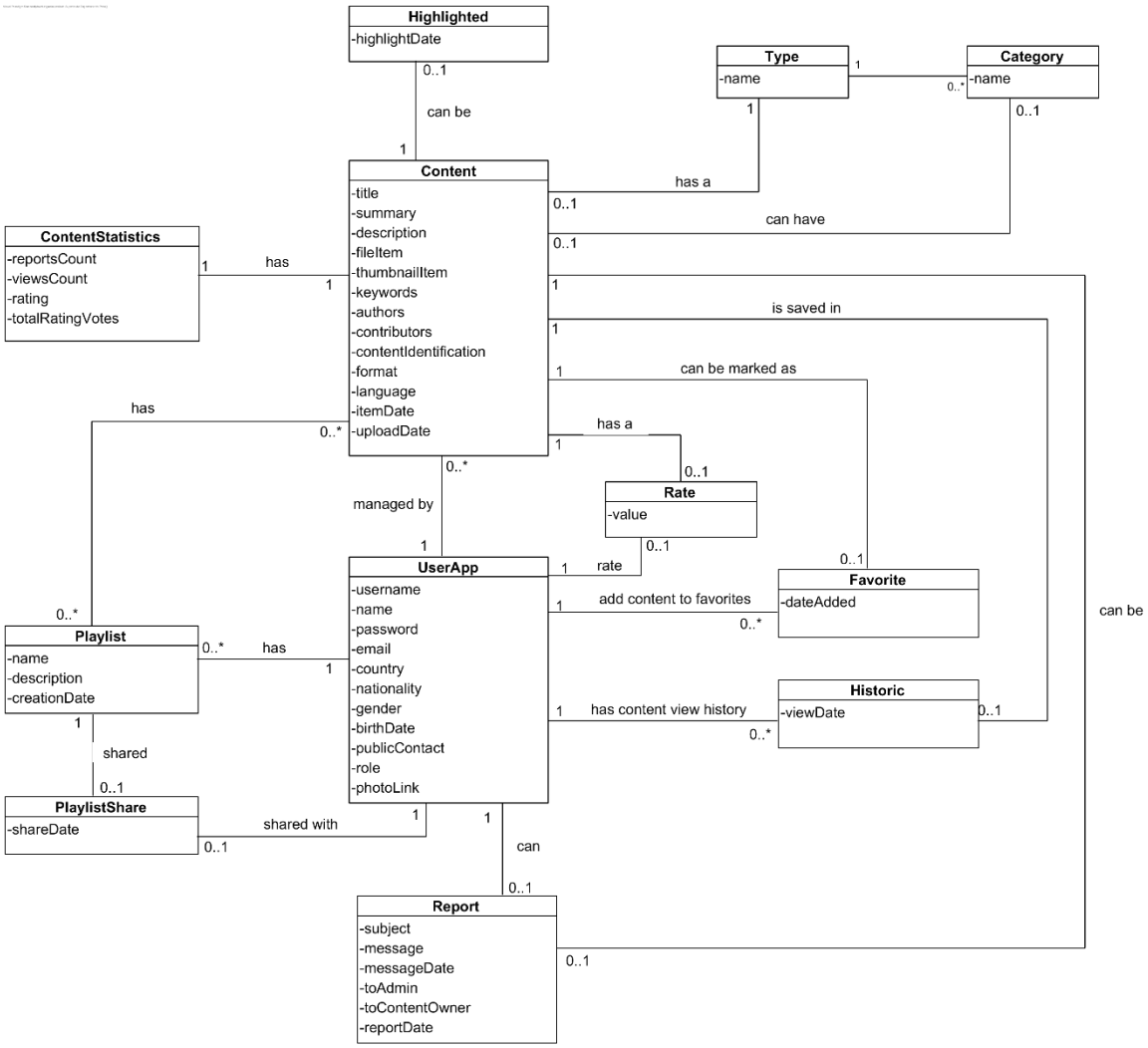



Figura 32 – Modelo de Domínio do sistema

Tabela 22 – Catálogo de elementos do modelo de domínio

| Elemento | Descrição |
|---|---|
| Content | Entidade que representa um conteúdo |
| ContentStatistics | Entidade que representa as estatísticas de um conteúdo |
| UserApp | Entidade que representa o utilizador do sistema |
| Highlighted | Entidade que representa os conteúdos em destaque |
| Type | Entidade que representa o tipo de conteúdo |
| Category | Entidade que representa a categoria do conteúdo |

| Elemento | Descrição |
|---|---|
| *Playlist* | Entidade que representa uma *Playlist* do utilizador |
| *Playlist*Share | Entidade que representa uma *Playlist* partilhada com outro utilizador |
| Favorite | Entidade que representa um conteúdo marcado como favorito pelo utilizador |
| Historic | Entidade que representa um conteúdo visualizado pelo utilizador |
| Report | Entidade que representa uma denúncia de um conteúdo |
| Rate | Entidade que representa a classificação atribuída por um utilizador a um conteúdo |

Um utilizador pode criar, editar e remover conteúdos a si associados. Para além dos seus atributos, cada conteúdo existente no sistema tem associado uma entidade que representa as suas estatísticas, como o número de visualizações e *rating* do conteúdo. Um conteúdo tem obrigatoriamente um tipo, pode ter uma categoria e pode ser marcado como um conteúdo em destaque. O utilizador tem também a possibilidade de adicionar conteúdos aos favoritos, ficando estes associados a si para consulta. À medida que o utilizador navega na plataforma, o sistema regista num histórico todos os conteúdos por ele visualizados. Para cada conteúdo, o utilizador pode registar uma denúncia indicando o(s) destinatário(s) dessa denúncia. Para além da criação de conteúdos, o utilizador pode proceder à criação de *Playlists* e ir adicionando e removendo os conteúdos que desejar a uma dada *Playlist*. Uma *Playlist* pode ser partilhada com outros utilizadores.

## 4.3 Modelo de Dados

Na Figura 33 encontra-se representado o modelo de dados com a estrutura desenhada para a construção da solução.

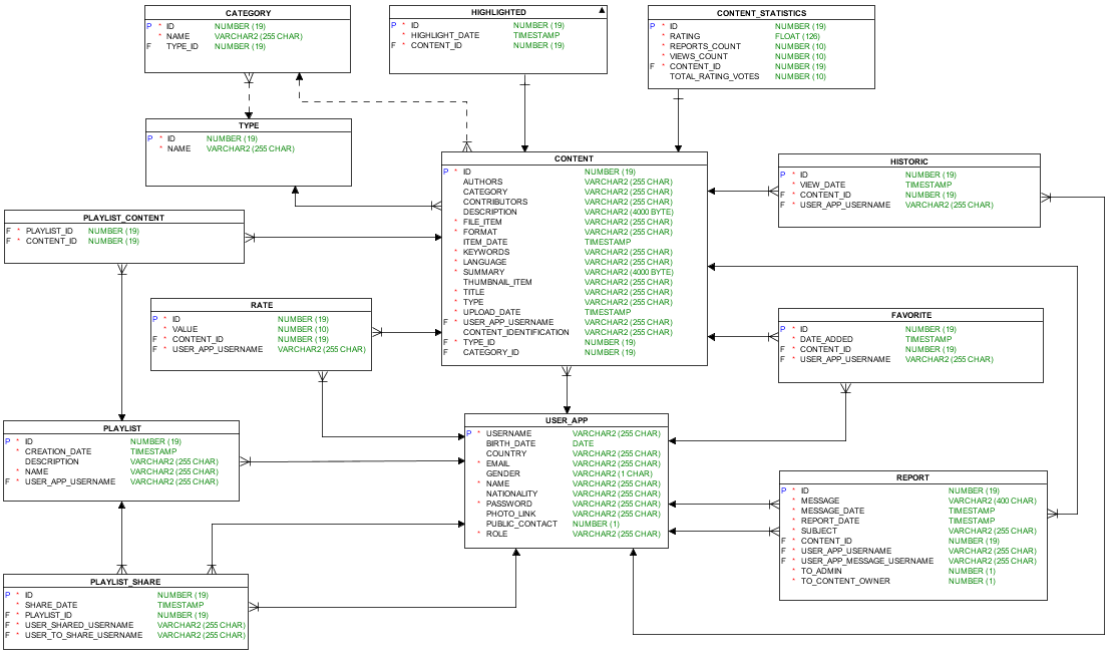



Figura 33 – Modelo de Dados

Tabela 23 – Catálogo de elementos do modelo de dados

| Elemento | Descrição |
|---|---|
| CONTENT | Tabela que armazena os conteúdos |
| CONTENT_STATISTICS | Tabela que armazena as estatísticas de um conteúdo |
| USER_APP | Tabela que armazena o utilizador registado no sistema |
| HIGHLIGHTED | Tabela que armazena os conteúdos em destaque |
| TYPE | Tabela que armazena os tipos de conteúdo existentes |
| CATEGORY | Tabela que armazena as categorias de conteúdo existentes |
| *PLAYLIST* | Tabela que armazena as *Playlists* |
| *PLAYLIST*_CONTENT | Tabela de associação entre *Playlist* e Content que armazena os conteúdos de cada *Playlist* |
| *PLAYLIST*_SHARE | Tabela que armazena as *Playlists* partilhadas com outro utilizador |
| RATE | Tabela que armazena a classificação atribuída por um utilizador a um conteúdo |
| FAVORITE | Tabela que armazena um conteúdo marcado como favorito pelo utilizador |

| Elemento | Descrição |
|---|---|
| HISTORIC | Tabela que armazena o histórico de visualizações de conteúdos de um utilizador |
| REPORT | Tabela que armazena o registo de uma denúncia de conteúdo efetuada pelo utilizador |

Como se pode visualizar na Figura 33, a estrutura do modelo de dados é semelhante ao modelo de domínio apresentado, uma vez que o modelo relacional da base de dados é gerado através do uso do JPA, que permite armazenar entidades de negócio como entidades relacionais. Um conteúdo (tabela "CONTENT") é criado por um utilizador (tabela "USER"), pelo que armazena uma chave estrangeira representativa do utilizador que consiste no seu *username* (coluna "USER_APP_USERNAME"). Um utilizador pode ter a si associado um ou mais conteúdos, e a cada conteúdo corresponde obrigatoriamente um registo na tabela de estatísticas (tabela "CONTENT_STATISTICS"). A ligação entre um conteúdo e as suas estatísticas é feita através da chave primária do conteúdo, armazenada como chave estrangeira (coluna "CONTENT_ID") nos registos da tabela "CONTENT_STATISTICS". Um conteúdo tem um tipo (tabela "TYPE") e pode ter uma categoria (tabela "CATEGORY"), efetuando a ligação a estes elementos assumindo as suas chaves primárias como chaves estrangeiras ("TYPE_ID" e "CATEGORY_ID").

A tabela "HIGHLIGHTED" é uma tabela responsável por armazenar os registos correspondentes aos conteúdos em destaque, armazenando, por isso, a chave primária de cada conteúdo destacado.

A tabela "TYPE" é uma tabela que regista os tipos de conteúdo existentes e a tabela "CATEGORY" armazena as categorias de conteúdo existentes. Um tipo pode conter várias categorias, pelo que cada categoria registada contém como chave estrangeira o tipo a que está associada.

Na base de dados são também registados os conteúdos favoritos de um utilizador (tabela "FAVORITE"). Um utilizador pode adicionar vários conteúdos aos favoritos e, por isso, cada registo nesta tabela é associado a um conteúdo e a um utilizador através da referência às chaves primárias dessas tabelas ("CONTENT_ID" e "USER_APP_USERNAME").

Com uma estrutura semelhante à tabela dos favoritos, existe uma tabela ("HISTORIC") para registar o histórico de visualização de conteúdos de um utilizador. Cada entrada nesta tabela corresponde a um utilizador e a um conteúdo visualizado, pelo que é adotada a mesma estratégia de referência às chaves primárias do utilizador e do conteúdo.

A tabela "*PLAYLIST"* é a tabela destinada a suportar a funcionalidade de gestão de *Playlists*. Um utilizador pode criar várias *Playlists* e associar diversos conteúdos a essas *Playlists*. Dado que uma *Playlist* pode conter vários conteúdos e um conteúdo pode pertencer a várias *Playlists*, verifica-se uma associação de muitos para muitos e, portanto, é usada uma tabela associativa entre as tabelas "*PLAYLIST"* e "CONTENT" com as suas chaves primárias como chave composta

da tabela de associação "*PLAYLIST*_CONTENT". De forma a saber a que utilizador pertence uma dada *Playlist*, é usado como referência uma chave estrangeira na tabela *Playlist* correspondente ao utilizador. Um utilizador pode efetuar a partilha de uma *Playlist* com outro utilizador, pelo que a tabela "*PLAYLIST*_SHARE" armazena a *Playlist* partilhada, o utilizador que a partilhou e o utilizador com quem foi partilhada a *Playlist*, registando para isso como chaves estrangeiras as chaves primárias dessas colunas.

## 4.4 Interface gráfica

Relativamente à interface gráfica a adotar na plataforma foram realizadas algumas maquetes como referência base para o desenvolvimento do *layout* da aplicação. Dado que a construção desta nova plataforma na ALERT surgiu fruto de um projeto anterior que partilhava um objetivo semelhante, existiam já algumas bases e ideias definidas para a interface do produto. Sobre a existência e posicionamento de certos componentes também se aproveitou o estudo do estado da arte efetuado, como, por exemplo, o posicionamento de uma área de pesquisa na barra de navegação e o esquema de pesquisa por facetas na página de pesquisa avançada. Para além disso, visto que a nova plataforma se insere na área de *training* da ALERT e tem como intuito servir de complemento ao ALERT® eLEARNING, há certos detalhes a ter em atenção de forma a manter a coerência entre os produtos desta área da empresa. Assim, destaca-se o facto da cor predominante dos botões e hiperligações ser o verde, cor essa que representa a área de *training* na ALERT, e a barra de navegação deve apresentar uma cor semelhante à cor adotada em áreas de navegação de outros produtos (cinzento escuro), como, por exemplo, no ALERT® ONLINE. Nas secções seguintes é apresentado com maior detalhe o planeamento efetuado para a interface gráfica das principais vistas da plataforma.

### 4.4.1 Cabeçalho e Rodapé

No cabeçalho da plataforma (Figura 34), comum a todas as páginas existentes, deve existir o logo desenhado pela ALERT para a biblioteca no canto superior esquerdo. Caso o utilizador esteja autenticado no sistema, no canto superior direito deve ser mostrada a sua foto de perfil.


Figura 34 – Cabeçalho da plataforma

No rodapé (Figura 35) de todas as páginas da plataforma deve existir uma referência para os direitos de autor e para as redes sociais da ALERT.


Figura 35 – Rodapé da plataforma

### 4.4.2 Barra de Navegação

Na Figura 36 é apresentado o desenho da página de visualização de conteúdos em destaque com o comportamento da barra de navegação e os elementos que a constituem.

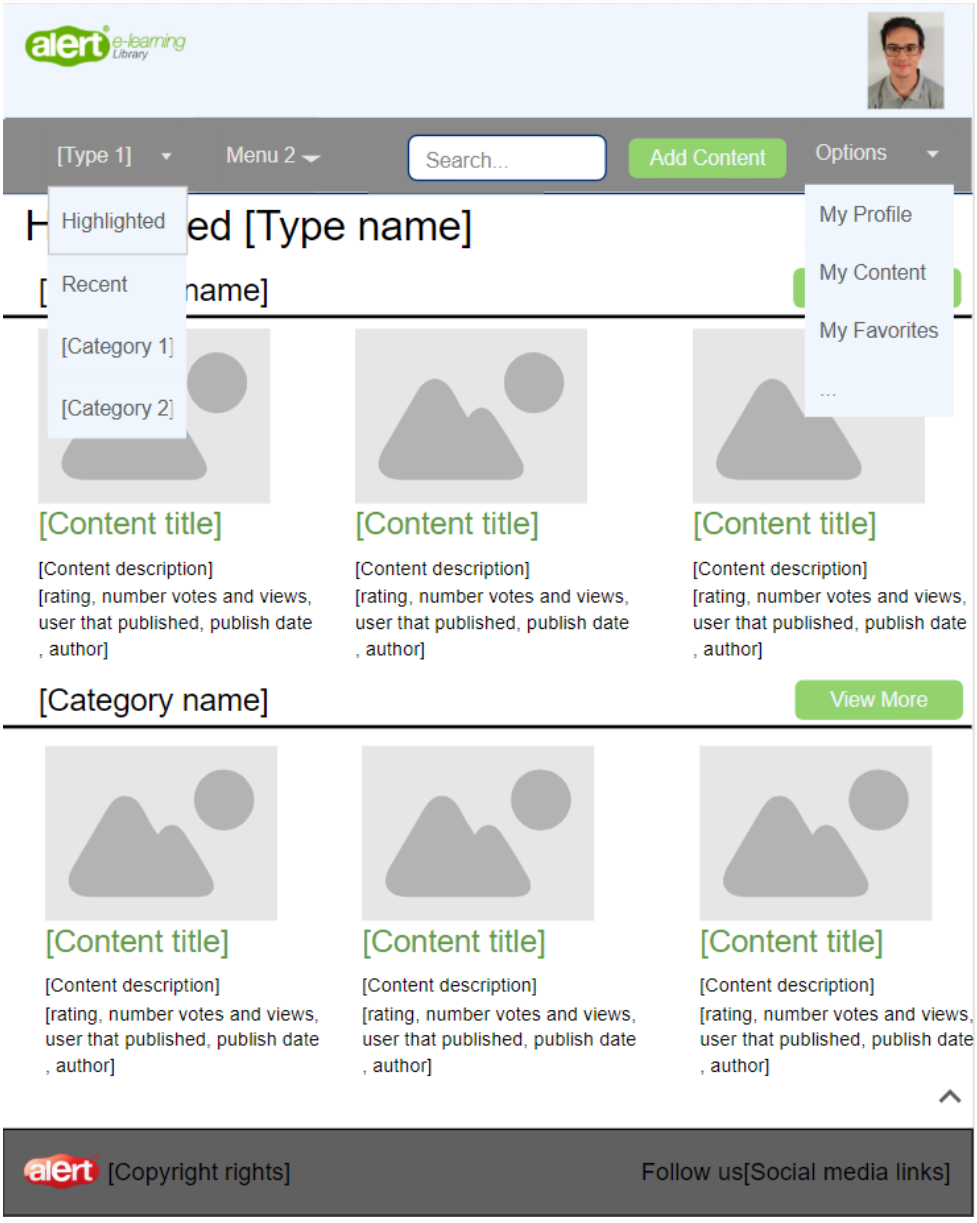



Figura 36 – Barra de navegação

A barra de navegação deve estar presente em todas as páginas da plataforma, apresentando os tipos de conteúdos existentes e as categorias de cada tipo sobre a forma de *dropdown* menus. Esta barra deve também conter uma caixa de pesquisa onde o utilizador possa inserir o texto pelo qual desejar pesquisar. Para além disso deve haver um atalho que destaque a função de publicação de conteúdo (caso o utilizador tenha permissões para tal) e um menu de acesso às opções existentes para o utilizador autenticado. Caso o utilizador não esteja autenticado, o atalho para a criação de conteúdo não deve aparecer e em vez do menu de opções deve surgir uma opção para a autenticação do utilizador.

### 4.4.3 Vista Principal de Conteúdos

Na Figura 37 é exposto o esboço da vista principal de conteúdos.

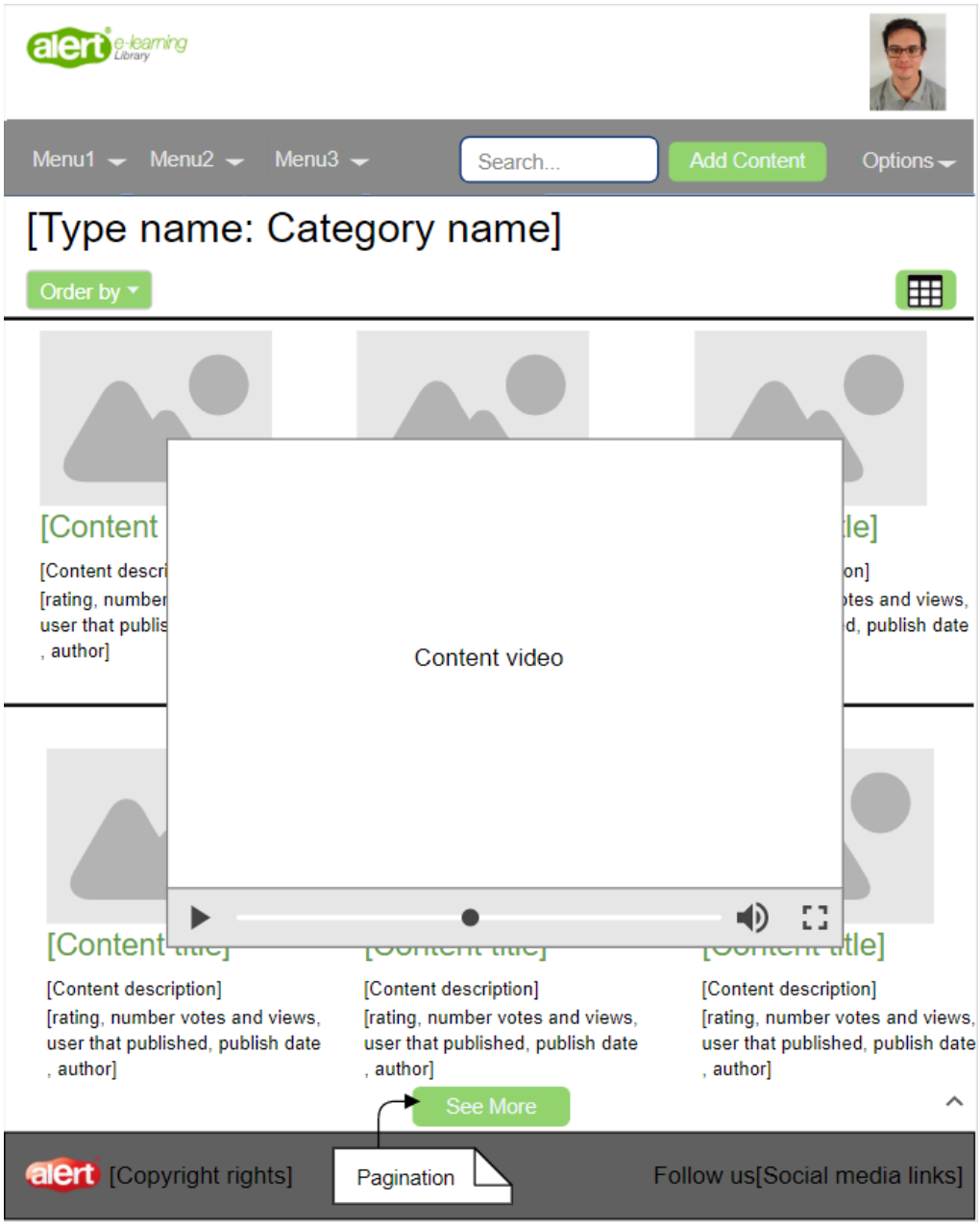



Figura 37 – Vista Principal de Conteúdos

A vista de conteúdos deve apresentar um formato comum para todas as páginas onde sejam apresentados diversos conteúdos. Esta vista (Figura 37) consiste numa vista em grelha, onde cada linha horizontal da página apresenta três conteúdos. Para cada conteúdo apresentado na página deve ser mostrada a sua *thumbnail,* o seu título e descrição, bem como a sua classificação, número de visualizações, o utilizador que o publicou e essa data de publicação, e o autor do conteúdo. Caso o número de conteúdos existentes para apresentar na página em navegação seja superior ao número de conteúdos já apresentados, deve existir uma opção de paginação após a última linha horizontal de conteúdos que carregue mais conteúdos para a página por baixo dos já existentes. Para além dessa opção, deve também existir a opção de

ordenação dos conteúdos pelos filtros disponibilizados e uma opção de mudança para uma vista em tabela dos conteúdos.

Ao selecionar a *thumbnail* de cada conteúdo, caso o conteúdo seja um vídeo, deve ser mostrada uma janela com o *video player* a reproduzir o vídeo. Caso o utilizador selecione o título do conteúdo ou o conteúdo esteja em formato PDF, o utilizador deve ser redirecionado para a página de detalhes do conteúdo.

### 4.4.4 Vista em Tabela de Conteúdos

Na Figura 38 é apresentada outra vista dos conteúdos existentes, mas no formato de vista em tabela.

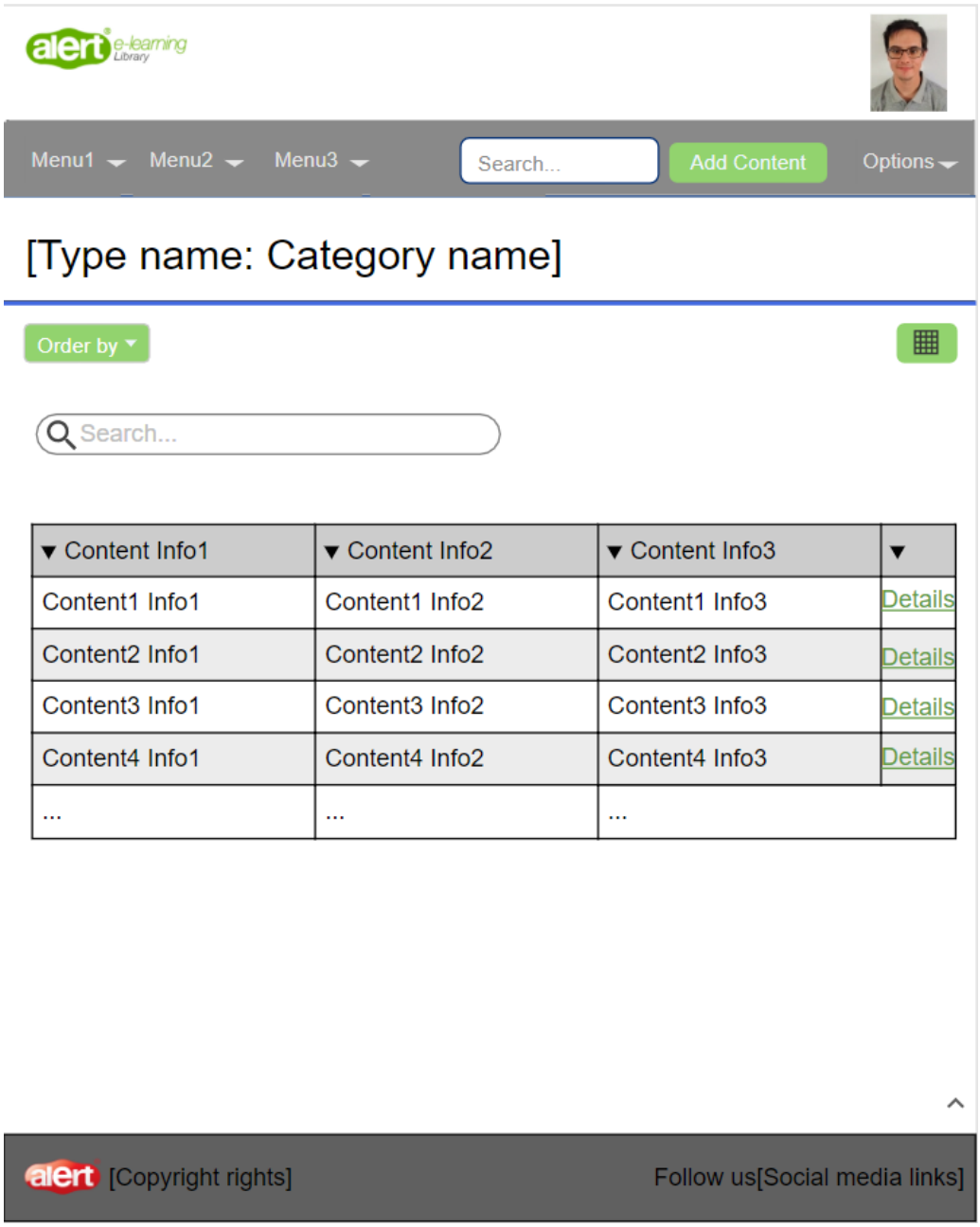



Figura 38 – Vista em Tabela de Conteúdos

Como alternativa à vista principal de conteúdos (vista em grelha), deve existir a acompanhá-la uma opção para a vista de conteúdos em tabela. Tal como na vista principal, deve existir uma opção de ordenação dos conteúdos, uma opção de voltar para a vista em grelha e uma opção de paginação por baixo da tabela para carregar mais conteúdos para a página caso seja necessário. Adicionalmente, nesta vista deve existir uma área para pesquisa por texto que filtra os resultados apresentados na tabela. Na tabela deve ser mostrado o título, tipo e categoria do conteúdo, data de publicação, classificação e número de visualizações. A cada conteúdo deve estar associada uma opção para aceder à sua página de detalhes.

### 4.4.5 Vista de Detalhes de Conteúdo

Na Figura 39 é apresentado o desenho da vista de detalhes de um conteúdo.

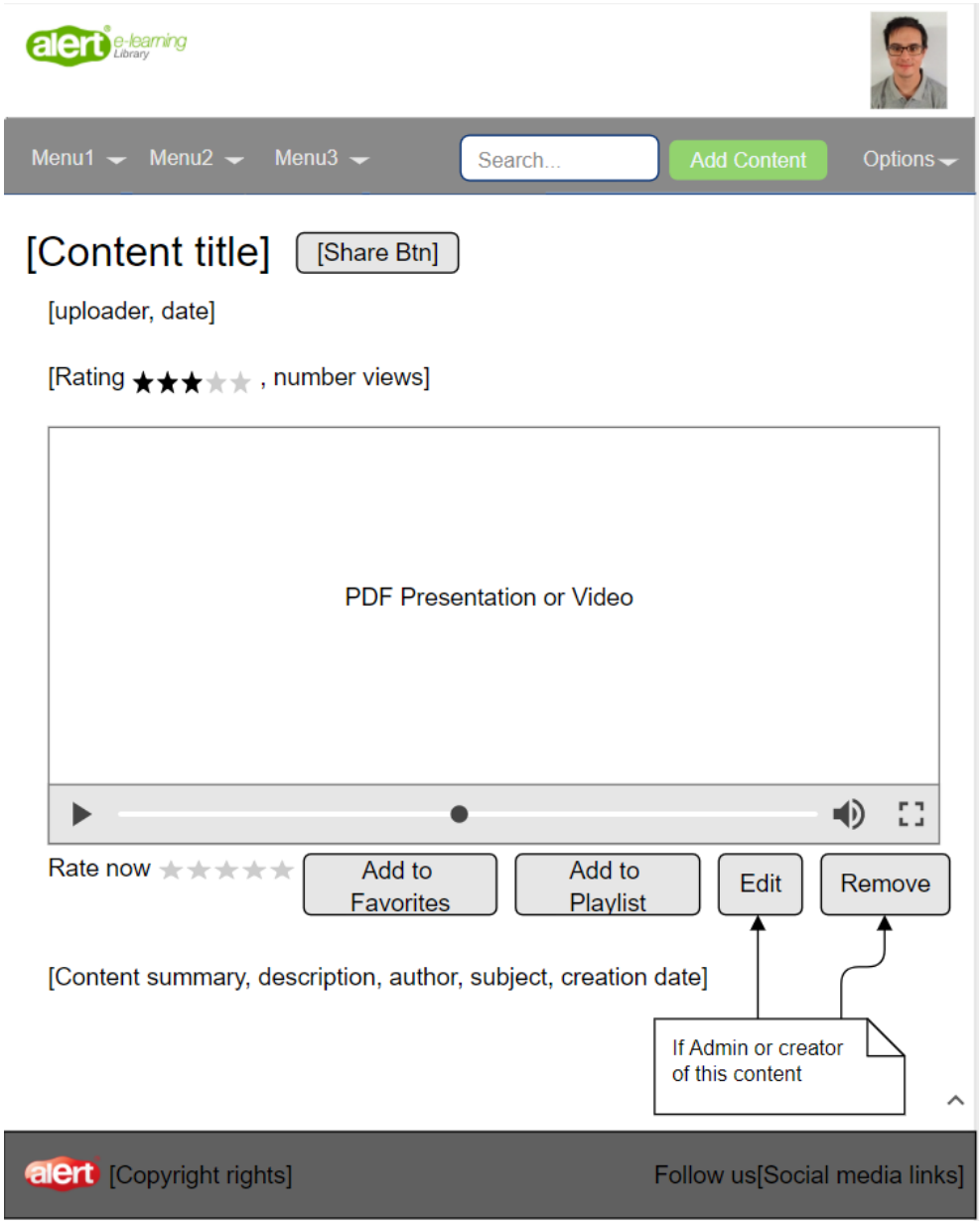



Figura 39 – Vista de Detalhes de Conteúdo

A página de detalhes de conteúdo deve apresentar os metadados do conteúdo em visualização. Assim, deve ser mostrado o seu título, o utilizador que o publicou e a data de publicação, a sua classificação e número de visualizações, a sua descrição, autor, palavras-chave e data de criação. Na área central da página deve ser apresentado o conteúdo (vídeo ou apresentação PDF) com as opções de lhe atribuir uma classificação, adicioná-lo aos favoritos e/ou a uma

*Playlist* e editá-lo ou removê-lo. Estas três primeiras opções devem ser apenas visíveis para os utilizadores autenticados, enquanto as duas últimas apenas para o utilizador autenticado que publicou o conteúdo na plataforma ou para um utilizador que seja administrador.

### 4.4.6 Conteúdos Destacados/Recentes

Na Figura 40 é apresentado o esboço para a vista dos conteúdos em destaque, semelhante à vista dos conteúdos mais recentes.

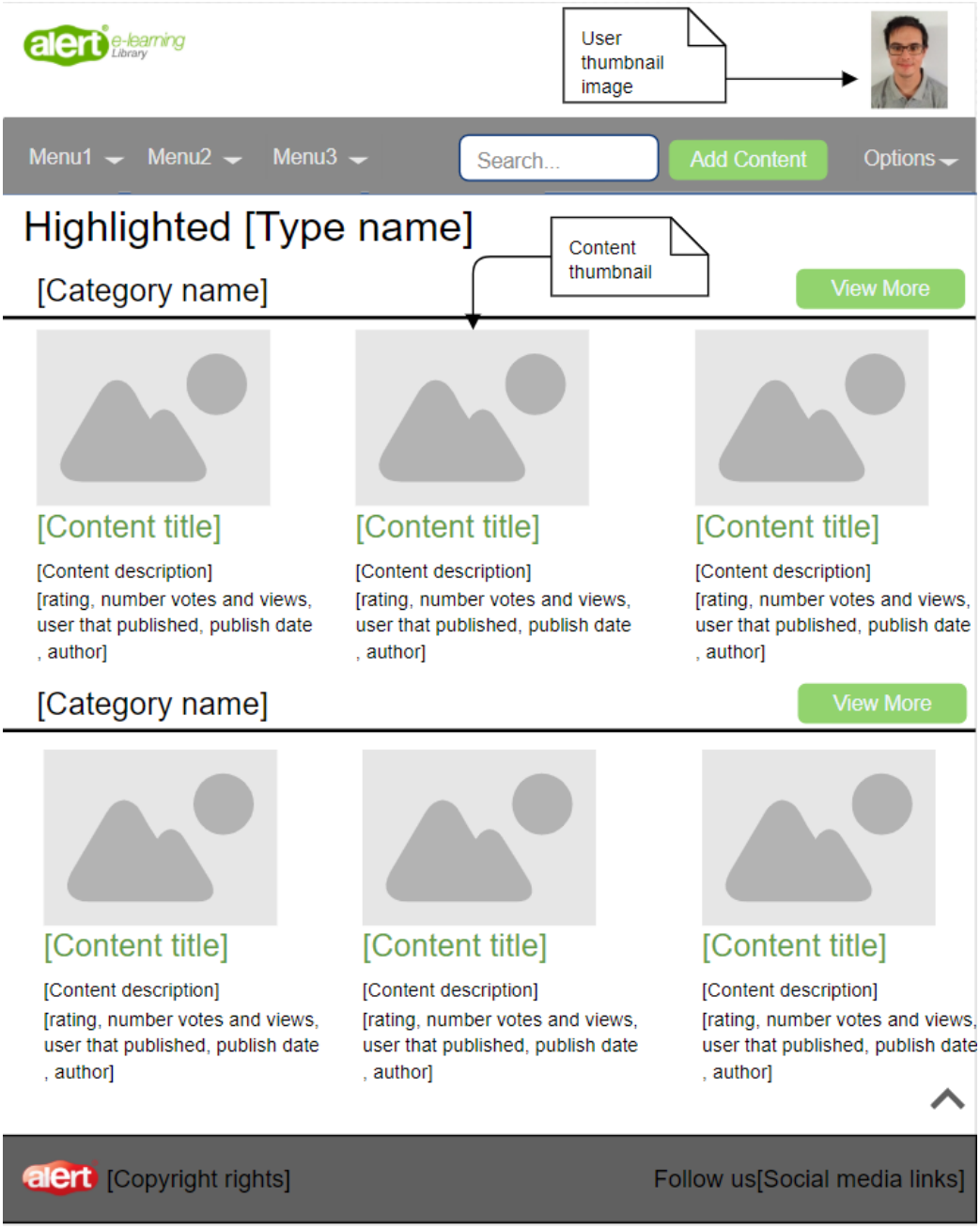



Figura 40 – Vista de Conteúdos em Destaque

Para cada tipo de conteúdo deve existir uma vista de conteúdos destacados e outra de conteúdos mais recentes. Quando o tipo de conteúdo em visualização possui várias categorias, devem ser apresentados os conteúdos em destaque e os mais recentes divididos por cada categoria do tipo em visualização. Para cada categoria deve ser disponibilizado um atalho para visualizar todos os conteúdos dessa categoria. Caso o tipo de conteúdo não tenha nenhuma categoria, são apresentados os conteúdos em destaque/recentes desse tipo sem a divisão por categorias.

### 4.4.7 Conteúdos do utilizador

Na Figura 41 é apresentada a vista planeada para a página de conteúdos de um utilizador.

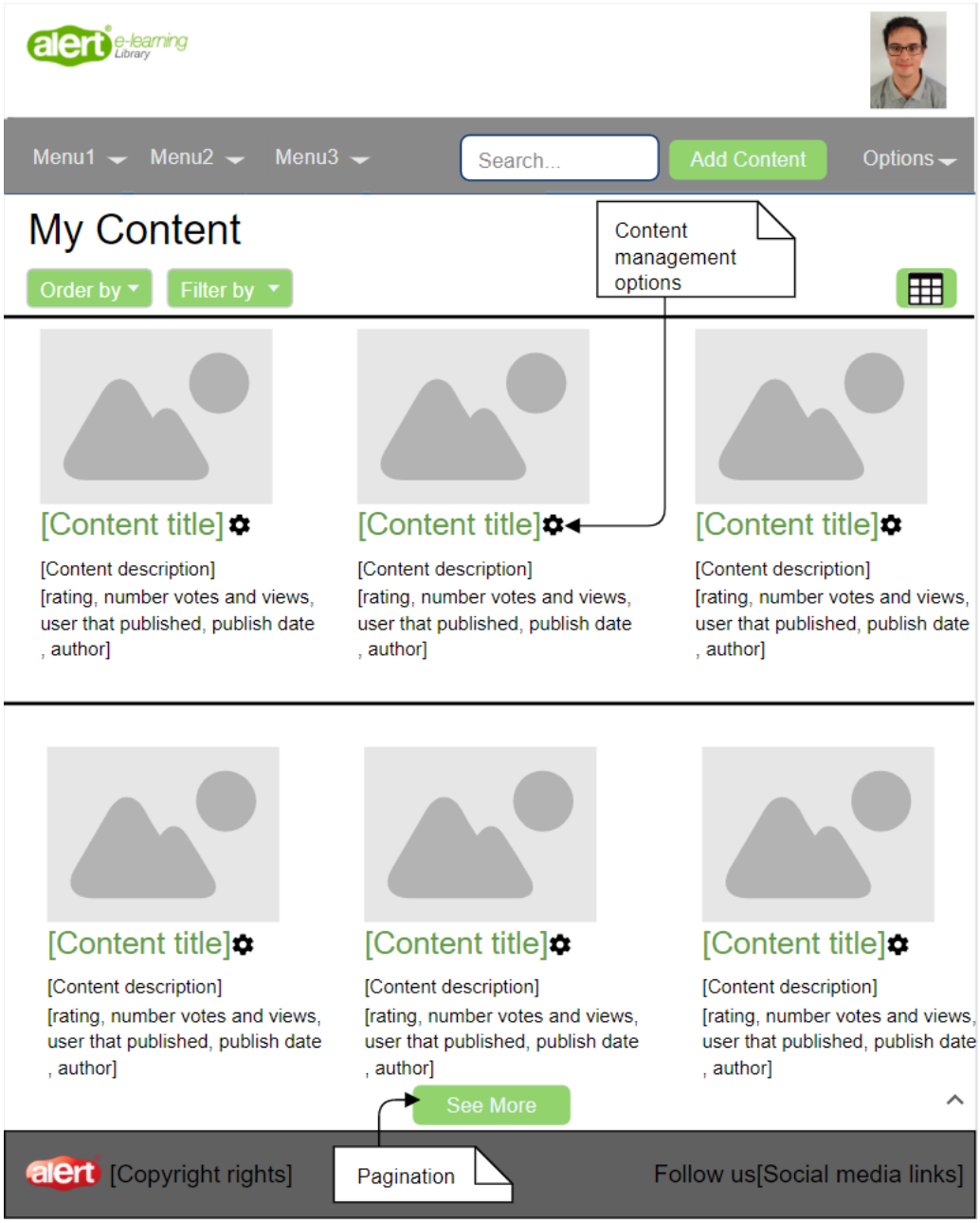



Figura 41 – Vista de Conteúdos do Utilizador

A vista principal de conteúdos publicado pelo próprio utilizador deve seguir o mesmo formato comum à vista de conteúdos apresentada na Figura 37, diferenciando-se em dois pontos: a opção de filtrar os conteúdos apresentados por tipo de conteúdo e o acesso às opções de gestão de cada conteúdo (editar e eliminar). Deve existir também a opção de vista em tabela semelhante ao formato comum, também com as duas diferenças anteriormente referidas.

As restantes áreas de visualização de conteúdos favoritos e histórico do utilizador devem ser semelhantes a este modelo, apresentando a vista em grelha e em tabela dos favoritos e do histórico de conteúdos consultados, diferenciando apenas nas opções de gestão apresentadas, onde só deve existir a opção de remover o conteúdo dos favoritos ou do histórico de visualização.

### 4.4.8 Pesquisa de Conteúdo

Na figura que se segue (Figura 42) é apresentado o modelo desenhado para a vista de pesquisa avançada de conteúdos.

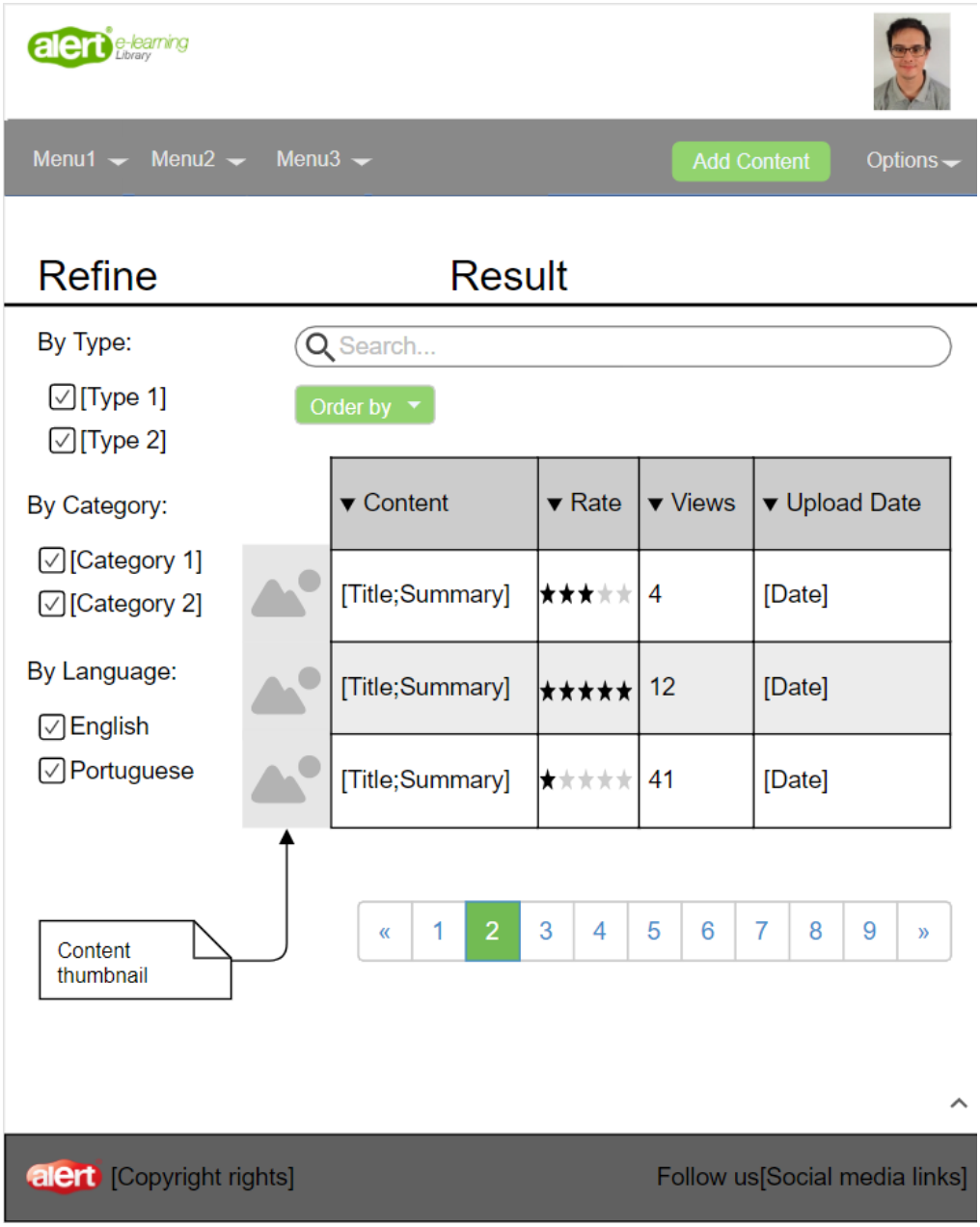



Figura 42 – Vista de Pesquisa de Conteúdo

A opção de pesquisa avançada de conteúdos deve apresentar uma área lateral onde seja possível efetuar uma pesquisa por facetas através da seleção de filtros existentes com base nas características dos conteúdos, como o tipo, categoria e língua dos conteúdos a pesquisar. Para cada filtro disponível deve ser apresentado o número de conteúdos existentes que correspondem a essa faceta. Na restante página deve estar disponível a opção de pesquisa por texto e, por baixo, devem ser mostrados os resultados da pesquisa, apresentando a *thumbnail* do conteúdo, o seu título e descrição, a sua classificação e número de visualizações e a sua data de publicação. Devem também existir as opções de ordenação e paginação dos resultados.

### 4.4.9 Criação de Conteúdo

Na Figura 43 é exposto o desenho da vista de criação de conteúdo.

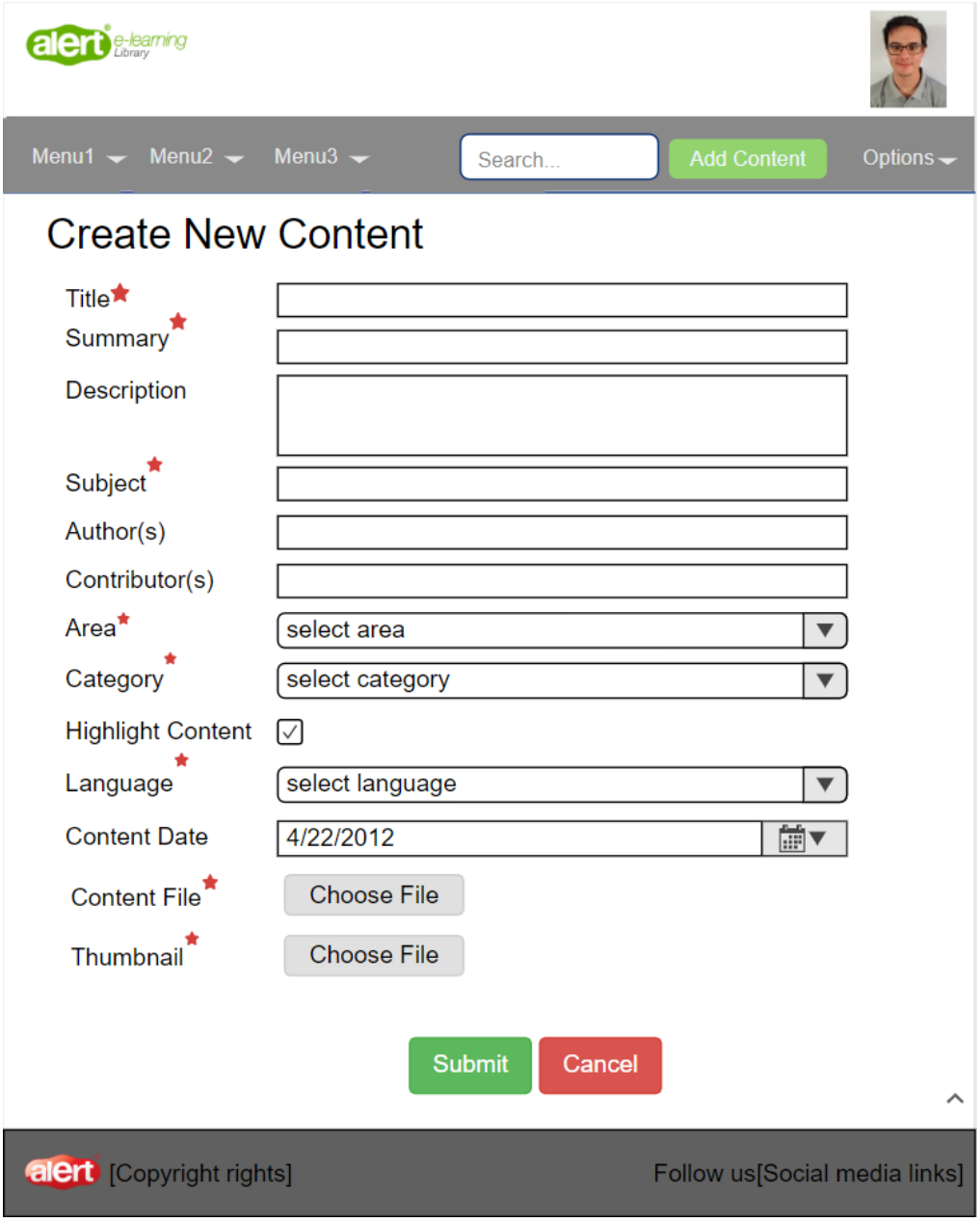



Figura 43 – Vista de Criação de Conteúdo

A vista de criação de conteúdo deve apresentar descritivos com campos de texto para o preenchimento dos dados necessários para a criação de um conteúdo. Para além dos campos de texto, os campos de seleção da área, categoria e língua do conteúdo devem ser em formato dropdown *list* para escolha do utilizador. Deve existir uma opção (*checkbox*) para destacar o conteúdo na área e categoria selecionadas e a inserção da data de criação do conteúdo deve ser assistida pelo sistema. Por fim, devem existir opções para selecionar o ficheiro do conteúdo e o ficheiro da *thumbnail* do conteúdo a ser criado e os botões para a submissão ou cancelamento da criação do conteúdo.

### 4.4.10 *Playlists*

Na Figura 44 é apresentado o modelo desenhado para a vista de *Playlists*.

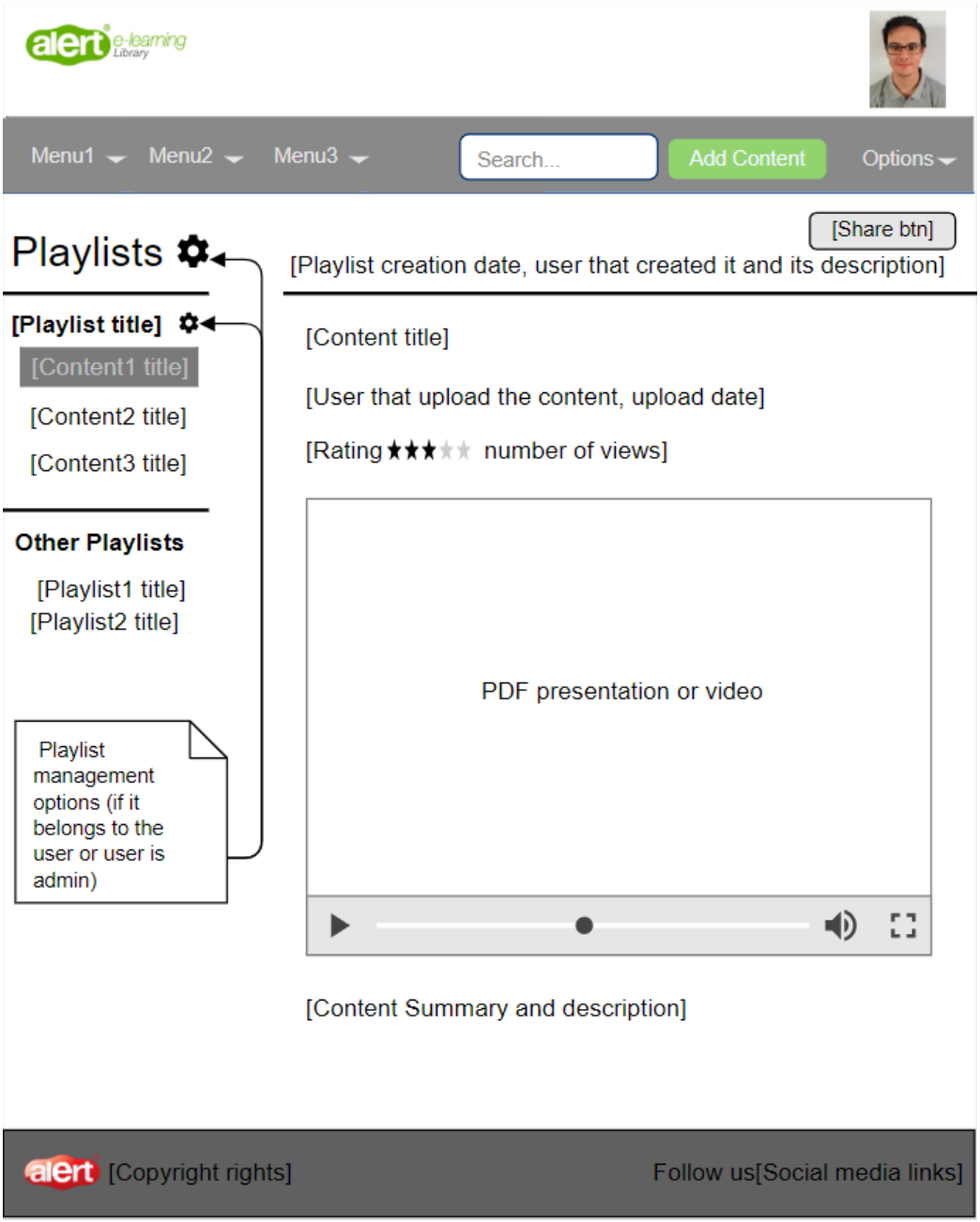



Figura 44 – Vista de *Playlists*

Na vista de *Playlists* deve ser apresentado uma área de navegação no canto esquerdo do ecrã que permita navegar pelos conteúdos da *Playlist* selecionada e selecionar outras *Playlists*. Na restante área do ecrã devem ser apresentados os detalhes da *Playlist* numa parte superior de destaque com a opção de partilha dessa *Playlist*, enquanto o conteúdo selecionado e a sua

informação deve ser mostrada abaixo dessa área. Se o utilizador autenticado for o criador da *Playlist* devem aparecer as opções de gestão de *Playlists* (adicionar, editar e remover *Playlist*) e de gestão de conteúdos pertencentes à *Playlist* (adicionar ou remover conteúdos da *Playlist*).

### 4.4.11 Perfil de utilizador

Na Figura 45 é exposto o desenho efetuado para a vista de perfil de utilizador.

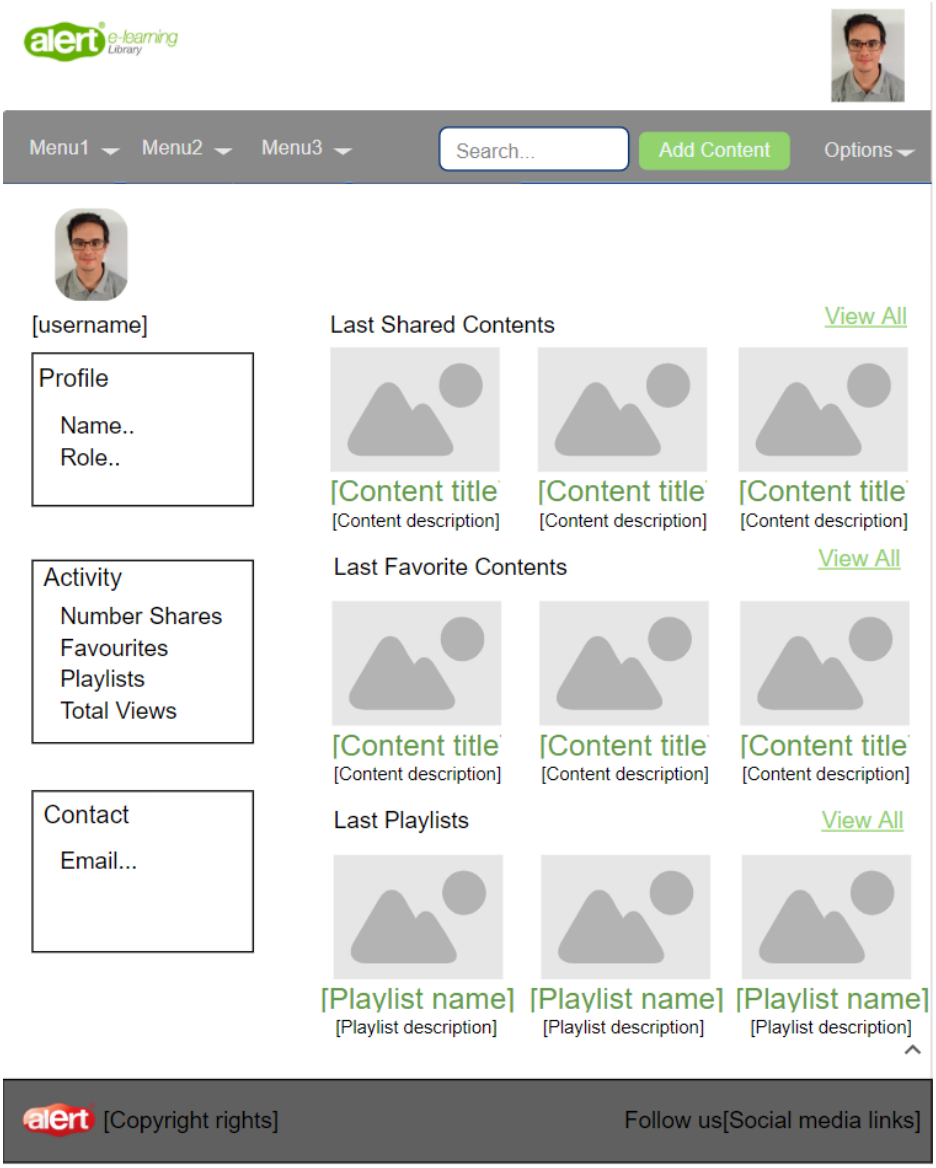



Figura 45 – Vista de Perfil de Utilizador

A vista de perfil de utilizador deve apresentar duas áreas distintas: no canto esquerdo do ecrã, deve ser mostrada a imagem do utilizador, algumas informações sobre o seu perfil, estatísticas da sua atividade na plataforma e o seu contacto; na restante área do ecrã devem ser destacados os últimos conteúdos publicados pelo utilizador, os seus últimos conteúdos marcados como

favoritos e as suas últimas *Playlists* criadas. Para cada um destes destaques, deve ser apresentado um atalho para visualizar todos os conteúdos, favoritos ou *Playlists* do utilizador.

# 5 Arquitetura

Neste capítulo são apresentados os artefactos relativos à arquitetura da solução implementada, exibindo a arquitetura do sistema através do diagrama de componentes e do diagrama de implantação.

## 5.1 Diagrama de Componentes

No diagrama de componentes exposto na Figura 46, pode-se verificar que a solução desenhada é constituída por quatro componentes numa arquitetura orientada a serviços REST. Este estilo arquitetural é usado para desenhar aplicações com baixo acoplamento através do protocolo HTTP, frequentemente usado no desenvolvimento de serviços web. Optou-se por esta arquitetura uma vez que possui um estilo arquitetural com um conjunto de restrições e práticas que contribuem para requisitos não funcionais desejáveis como: desempenho, escalabilidade, simplicidade, modificabilidade, visibilidade, portabilidade e confiabilidade (Fielding, 2000). Devido à separação de preocupações e à modularidade desta arquitetura, garante-se que no futuro possam ser acrescentadas funcionalidades e componentes de forma mais fácil e prática. Para além disso, verificou-se que alguns dos sistemas de *software* de Bibliotecas Digitais estudados na secção 2.2 também apresentam uma arquitetura deste tipo.

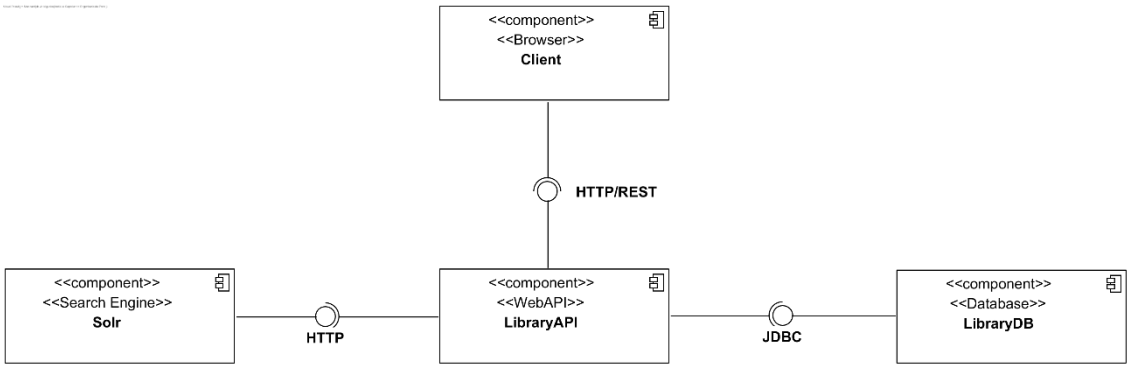



Figura 46 - Diagrama de Componentes do sistema

Tabela 24 – Catálogo de elementos do diagrama de componentes

| Elemento | Descrição |
|---|---|
| LibraryAPI | Representa o componente correspondente à API do sistema |
| Client | Componente que representa o dispositivo cliente que através de um navegador de internet (*browser*) consome os serviços da LibraryAPI |
| LibraryDB | Componente que representa a base de dados da aplicação |
| Solr | Componente que representa o motor de pesquisa utilizado |

No centro dessa arquitetura, como componente principal, encontra-se a "LibraryAPI", uma RESTful API desenvolvida com Spring Framework que contém a lógica do sistema e é responsável pelas principais operações, disponibilizando pontos de acesso aos serviços *web* através do protocolo de comunicação HTTP. O componente "*Client*" representa o dispositivo responsável por apresentar a interface gráfica ao utilizador através de um navegador de internet, estabelecendo assim a comunicação com os serviços da *Web Api*. Desta forma, é adotado o modelo Cliente-Servidor.

A "LibraryAPI" é responsável por comunicar com o componente "LibraryDB" que efetua o armazenamento dos dados na base de dados. A "LibraryAPI" utiliza a *framework* Hibernate e implementa JPA (*Java Persistence API*). O JPA é uma API padrão da linguagem Java que descreve uma interface comum para *frameworks* de persistência de dados, permitindo armazenar entidades de negócio como entidades relacionais, definindo um meio de mapeamento objeto-relacional (International Business Machines, 2019). Como mencionado nas seções de restrições de desenho e requisitos de implementação (4.1.2.5 e 4.1.2.6 respetivamente), a API está restringida ao uso de linguagem Java e a base de dados ao sistema de gestão de base de dados Oracle.

Para além do armazenamento de dados na base de dados relacional, a "LibraryAPI" é também responsável por gerir os dados inseridos no motor de pesquisa Solr e manter a sincronização entre estes dados e os dados da base de dados, consumindo os serviços do motor de pesquisa através da SolrJ. A SolrJ é uma API que facilita a comunicação das aplicações Java com o Solr, estabelecendo comunicações via HTTP. Optou-se pelo motor de pesquisa Solr com base no estudo e comparações efetuadas nas seções 2.3 e 3.4.1 e também porque é um componente já utilizado na ALERT noutros projetos.

## 5.2 Diagrama de Implantação

Na Figura 47 é apresentado o diagrama de implantação dos componentes do sistema.

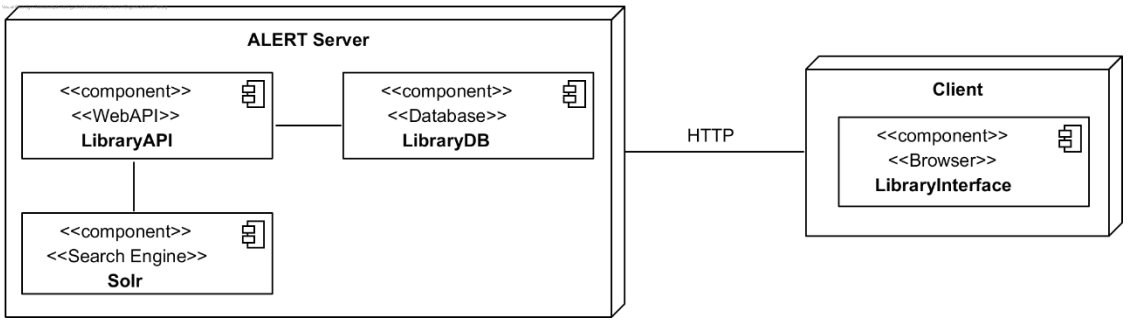



Figura 47 – Diagrama de Implantação

Tabela 25 – Catálogo de elementos do diagrama de implantação

| Elemento | Descrição |
|---|---|
| ALERT Server | Servidor Windows da ALERT |
| LibraryAPI | Representa o componente correspondente à API do sistema |
| LibraryInterface | Componente que representa um navegador de internet (*browser*) onde o utilizador interage com a interface gráfica e consome os serviços da LibraryAPI |
| LibraryDB | Representa a base de dados do sistema |

A "LibraryAPI", o motor de pesquisa Solr e a base de dados encontram-se alojadas num servidor Windows disponibilizado pela ALERT, sendo que a API está a ser executada no Apache Tomcat. A aplicação *web* desenvolvida que corresponde à interface com que o utilizador interage ("LibraryInterface") é acedida através de um navegador de internet na máquina do utilizador e efetua as trocas de comunicação com a "LibraryAPI" através do protocolo HTTP.

# 6 Desenho e Implementação

Neste capítulo é exposto a estrutura da solução desenvolvida e o desenho e a implementação das principais funcionalidades do sistema, tendo em conta a análise e arquitetura previamente definidas para a construção da solução, apresentando evidências da implementação efetuada com as respetivas explicações.

## 6.1 Estrutura

Na implementação da "LibraryAPI" procurou-se seguir as boas práticas de Engenharia de *Software* através da adoção de uma arquitetura que possibilitasse pôr em prática diversos padrões de *software*. Assim, a estrutura da API desenvolvida é apresentada na Figura 48.

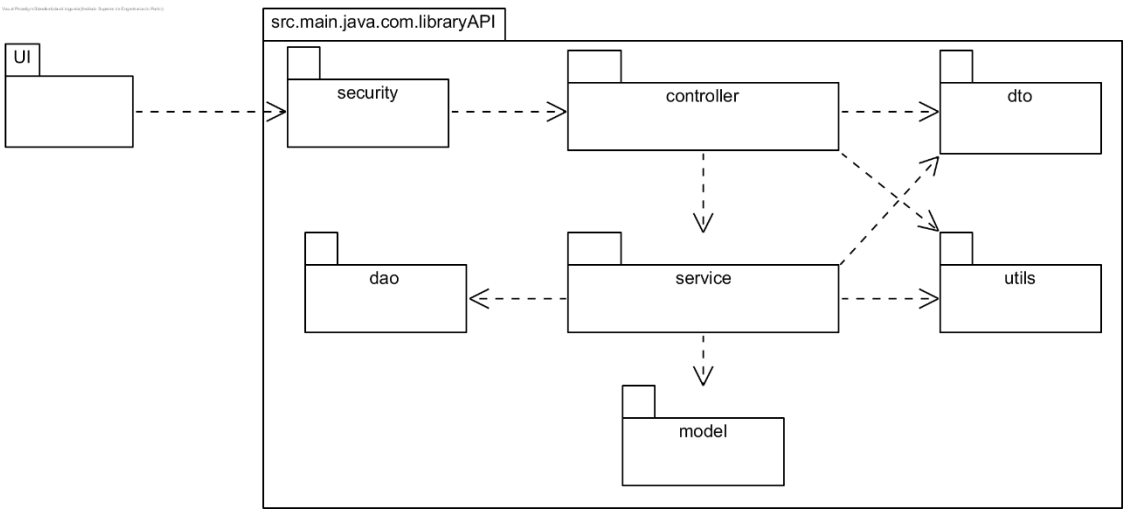



Figura 48 – Estrutura da LibraryAPI

Como se pode ver na figura anterior (Figura 48), o projeto da “LibraryAPI” foi desenvolvido segundo o padrão de Camadas resultando na seguinte estrutura:

- “controller”: esta camada agrega todos os controladores da API que têm a responsabilidade de receber e controlar os pedidos do exterior, delegando as operações necessárias a realizar para outra camada do sistema. Desta forma, é implementado o padrão *Controller*;

- “model”: camada que contém objetos necessários para operações de negócio e classes de persistência para o mapeamento objeto-relacional;

- “service": camada que executa as operações de negócio necessárias sobre os pedidos recebidos no controlador e comunica com a camada de persistência. De forma a isolar os detalhes de implementação são declaradas interfaces para cada serviço com os métodos pretendidos e, nas classes de implementação desses serviços, é então efetuada a implementação dos métodos;

- *Data Access Object* (“dao”): padrão que isola a camada de negócio da camada de persistência. Nesta camada de persistência foram criadas interfaces que comunicam com a base de dados extendendo a interface “JpaRepository”;

- *Data Transfer Object* (“dto”): padrão usado para transferir dados entre processos num objeto. Esta camada contém todos os objetos usados para agilizar a troca de dados entre o cliente e a “LibraryAPI”;

- “Security”: camada que contém as classes responsáveis por configurações e verificações de segurança da “LibraryAPI”;

- “Utils”: camada que contém todas as classes com operações úteis e auxiliares que são facilmente reutilizáveis por qualquer método.

Ao implementar esta estrutura são respeitadas boas práticas de desenvolvimento de *software* ao adotar padrões como alta coesão, baixo acoplamento, indireção e princípo da responsabilidade única.

## 6.2 Autenticação e Autorização

O processo de autenticação na plataforma desenvolvida passa por comparar as credenciais de acesso inseridas pelo utilizador que se pretende autenticar com as credenciais existentes no *Active Directory* (AD) da ALERT através do *Lightweight Directory Access Protocol* (LDAP). Desta forma os colaboradores da empresa conseguem autenticar-se na plataforma com as suas credenciais da ALERT e o sistema desenvolvido não tem a responsabilidade de manter os dados atualizados, sendo apenas necessário ter um utilizador registado na base de dados com o mesmo *username* usado na autenticação por LDAP. Para além disso, o uso deste protocolo

permite obter uma *thumbnail* para usar como foto de perfil do utilizador. Este processo de autenticação foi desenvolvido usando bibliotecas disponibilizadas pelo Spring Security. Depois da autenticação ser efetuada com sucesso, é gerado um *JSON Web Token* (JWT) que é devolvido e guardado no lado cliente de forma a ser usado para validar a autorização de acesso do utilizador nos pedidos seguintes. Na Figura 49 é apresentado um esquema dos processos de autenticação e autorização implementados seguido de uma descrição detalhada dos mesmos.

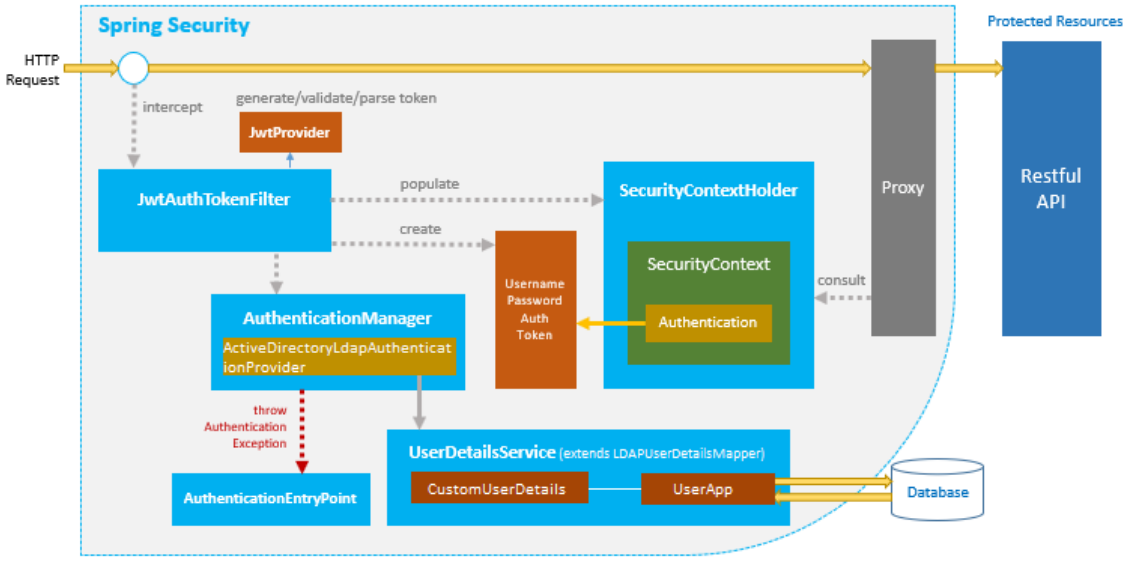



Figura 49 – Processo de Autenticação e Autorização. Adaptado de (grokonez, 2018)

Quando é efetuado um pedido HTTP para a "LibraryAPI", este passa por uma série de filtros relativos aos processos de autenticação e autorização. Na classe "JwtAuthTokenFilter" é feito o pré-processamento do pedido HTTP e espoletadas diferentes ações, dependendo da necessidade de autenticação ou autorização.

No caso da autenticação, quando um utilizador tenta autenticar-se na plataforma, é criado um objeto "UsernamePasswordAuthenticationToken" com o *username* e *password* inseridos. Este objeto é necessário para a tentativa de autenticação, uma vez que é usado pelo método de autenticação do "AuthenticationManager". No Extrato de código 1 é apresentado o início do processo de autenticação, com a criação do objeto e invocação do método atrás referido.

```
@Override
    public Authentication attemptAuthentication(HttpServletRequest req,
                                                HttpServletResponse res)
throws AuthenticationException {
        try {
            UserApp creds = new
ObjectMapper().readValue(req.getInputStream(),UserApp.class);

            return authenticationManager.authenticate(
                    new UsernamePasswordAuthenticationToken(
                            creds.getUsername(),
                            creds.getPassword(),
                            new ArrayList<>())
```

```
            );
        } catch (IOException e) {
            throw new RuntimeException(e);
        }
    }
```

Extrato de código 1 – Início do processo de autenticação

O "AuthenticationManager" utiliza o "ActiveDirectoryLdapAuthenticationProvider" configurado para efetuar a conexão ao LDAP da ALERT e é validada a instância do "UsernamePasswordAuthenticationToken" com as credenciais do utilizador. Caso essa instância seja inválida, é lançada uma exceção de autenticação. Caso a instância seja validada com sucesso, é usado um método da classe "UserDetailsService" que tem acesso aos dados do utilizador obtidos do LDAP. Neste método é criada e retornada uma instância do objeto "CustomUserDetails", necessário para construir um objeto de autenticação ("Authentication"). Visto que a interface "LdapUserDetails" não armazena, por predefinição, a *thumbnail* do utilizador obtida através do LDAP, foi então criada a classe "CustomUserDetails". Esta é uma classe que implementa a interface "LdapUserDetails", contendo atributos para armazenar os dados obtidos do LDAP, mas com as modificações necessárias para também armazenar a *thumbnail*. De forma a obter a *role* do utilizador, é efetuado um pedido à base de dados com o seu *username*, sendo obtido uma instância do "UserApp" que contém a *role* necessária para popular o objeto "CustomUserDetails". No Extrato de código 2 é apresentado a implementação do método atrás referido, onde é retornada uma instância de "CustomUserDetails".

```
@Override
    public CustomUserDetails mapUserFromContext(DirContextOperations ctx,
String username, Collection<? extends GrantedAuthority> authorities) {
        LdapUserDetailsImpl details = (LdapUserDetailsImpl)
super.mapUserFromContext(ctx, username, authorities);
        CustomUserDetails customUserDetails = new CustomUserDetails();
        byte[] photo = null;
        try {
            photo = (byte[])
ctx.getAttributes().get("thumbnailPhoto").get();
        } catch (NamingException e) {
            e.printStackTrace();
        }
        UserApp applicationUser =
userRepo.findByUsernameIgnoreCase(username);
        if (applicationUser == null) {
            throw new UsernameNotFoundException(username);
        }
        customUserDetails.setDn(details.getDn());
        customUserDetails.setPassword(details.getPassword());
        customUserDetails.setUsername(details.getUsername());

customUserDetails.setAuthorities(AuthorityUtils.createAuthorityList(applica
tionUser.getRole()));
customUserDetails.setGraceLoginsRemaining(details.getGraceLoginsRemaining()
);
customUserDetails.setTimeBeforeExpiration(details.getTimeBeforeExpiration()
);
        customUserDetails.setThumbnailPhoto(photo);
        return customUserDetails;
```

```
    }
```

Extrato de código 2 – Criação do CustomUserDetails com dados obtidos do LDAP e da base de dados

Tendo o objeto de autenticação criado com todos os dados necessários, é então gerado o *token* de acesso com o *username* do utilizador, a sua *role* e o tempo de expiração do *token*. Este *token* é inserido no cabeçalho de resposta do pedido, juntamente com a *thumbnail* codificada em Base64 para ser interpretada pelo cliente. O método que trata de efetuar estas operações é apresentado no Extrato de código 3.

```
@Override
    protected void successfulAuthentication(HttpServletRequest req,
                                            HttpServletResponse res,
                                            FilterChain chain,
                                            Authentication auth) throws
IOException, ServletException {

        CustomUserDetails userDetails = (CustomUserDetails)
auth.getPrincipal();

        String token = JWT.create()
                .withClaim("authorities",
auth.getAuthorities().iterator().next().getAuthority().toString())
                .withSubject(userDetails.getUsername())
                .withExpiresAt(new Date(System.currentTimeMillis() +
EXPIRATION_TIME))
                .sign(HMAC512(SECRET.getBytes()));

        res.addHeader(HEADER_STRING, TOKEN_PREFIX + token);
        PrintWriter writer = res.getWriter();

writer.append(DatatypeConverter.printBase64Binary(userDetails.getThumbnailP
hoto()));
        writer.close();
 }
```

Extrato de código 3 – Criação e adição do *token* e da *thumbnail* ao cabeçalho da resposta

No caso da autorização, quando o utilizador tenta aceder a um *endpoint* da "LibraryAPI" com o seu *token*, a classe "JwtAuthTokenFilter" interceta o pedido e obtém o *token* do cabeçalho de autorização, validando-o e interpretando os dados nele contidos, nomeadamente o *username* e a sua *role*. Caso o *token* seja inválido, é enviada uma mensagem de erro. Caso seja válido, com os dados recolhidos é devolvida uma instância de "UsernamePasswordAuthenticationToken". No Extrato de código 4 é visível este processo de interpretação e recolha de dados.

```
private UsernamePasswordAuthenticationToken
getAuthentication(HttpServletRequest request) {
        String token = request.getHeader(HEADER_STRING);
        if (token != null) {

            String user = JWT.require(Algorithm.HMAC512(SECRET.getBytes()))
                    .build()
                    .verify(token.replace(TOKEN_PREFIX, ""))
                    .getSubject();

            Claim claim = JWT.require(Algorithm.HMAC512(SECRET.getBytes()))
                    .build()
```

```
                    .verify(token.replace(TOKEN_PREFIX, ""))
                    .getClaim("authorities");

            if (user != null) {
                List<String> authorities = new ArrayList<String>();
                authorities.add(claim.asString());
                return new UsernamePasswordAuthenticationToken(user, null,
authorities.stream().map(SimpleGrantedAuthority::new).collect(Collectors.to
List()));
            }
            return null;
        }
        return null;
    }
```

Extrato de código 4 – Método de validação e recolha de dados do *token* de acesso

A instância retornada pelo método acima é então usada para criar e armazenar o objeto "Authentication" no objeto "SecurityContextHolder". O "SecurityContextHolder" é usado pelo Spring Security e é fundamental pois armazena os detalhes do contexto atual de segurança da plataforma relativamente ao utilizador, permitindo a consulta desses dados em qualquer local da "LibraryAPI". Depois do "SecurityContextHolder" estar definido com o objeto de autenticação, este é utilizado nos filtros de autorização existentes que validam se o utilizador tem então as permissões necessárias para aceder ao *endpoint* pretendido. No Extrato de código 5 é apresentado o método que recebe a instância gerada no Extrato de código 4 e é responsável pela definição do "SecurityContextHolder".

```
    @Override
    protected void doFilterInternal(HttpServletRequest req,
                                    HttpServletResponse res,
                                    FilterChain chain) throws IOException,
ServletException {
        String header = req.getHeader(HEADER_STRING);

        if (header == null || !header.startsWith(TOKEN_PREFIX)) {
            chain.doFilter(req, res);
            return;
        }
        try {
        UsernamePasswordAuthenticationToken authentication =
getAuthentication(req);

SecurityContextHolder.getContext().setAuthentication(authentication);
        } catch (Exception e) {
            res.sendError(HttpServletResponse.SC_UNAUTHORIZED, "Access
Denied");
            return;
        }
        chain.doFilter(req, res);
    }
```

Extrato de código 5 – Processo de autorização

As opções de acesso aos *endpoints* disponibilizados pela "LibraryAPI" são configuradas manualmente através da implementação do método "configure" da classe "WebSecurityConfigurerAdapter" do Spring Security. Para isso, é feita a indicação do método

HTTP do *endpoint* (get, post, put, delete), do URL do *endpoint* e do tipo de permissão de acesso que se pretende atribuir. No Extrato de código 6 são apresentados alguns exemplos das configurações de acesso definidas.

```
@Override
    protected void configure(HttpSecurity http) throws Exception {
        http.cors().and().csrf().disable().authorizeRequests()
        .antMatchers(HttpMethod.OPTIONS, "/**").permitAll()
        .antMatchers(HttpMethod.POST, SEARCH).permitAll()
        .antMatchers(HttpMethod.GET, GET_USER_PROFILE_DATA).permitAll()
        .antMatchers(HttpMethod.GET,
HIGHLIGHTED_CONTENTS).hasAnyAuthority("ADMIN","HIGH_USER")
        .antMatchers(HttpMethod.POST, SIGN_UP_URL).hasAnyAuthority("ADMIN")
        .anyRequest().authenticated()
        .and()
        .addFilter(new JWTAuthenticationFilter(authenticationManager()))
        .addFilter(new JWTAuthorizationFilter(authenticationManager()))
        .sessionManagement().sessionCreationPolicy(SessionCreationPolicy.ST
ATELESS);
 }
```

Extrato de código 6 – Configurações de segurança de acesso a *endpoints* da API

## 6.3 Criar Conteúdo

Nesta secção é descrito o processo de criação de conteúdo. De forma a facilitar a leitura dos diagramas, este processo é apresentado com recurso a dois diagramas de sequência (Figura 50 e Figura 51) seguido das suas respetivas explicações.

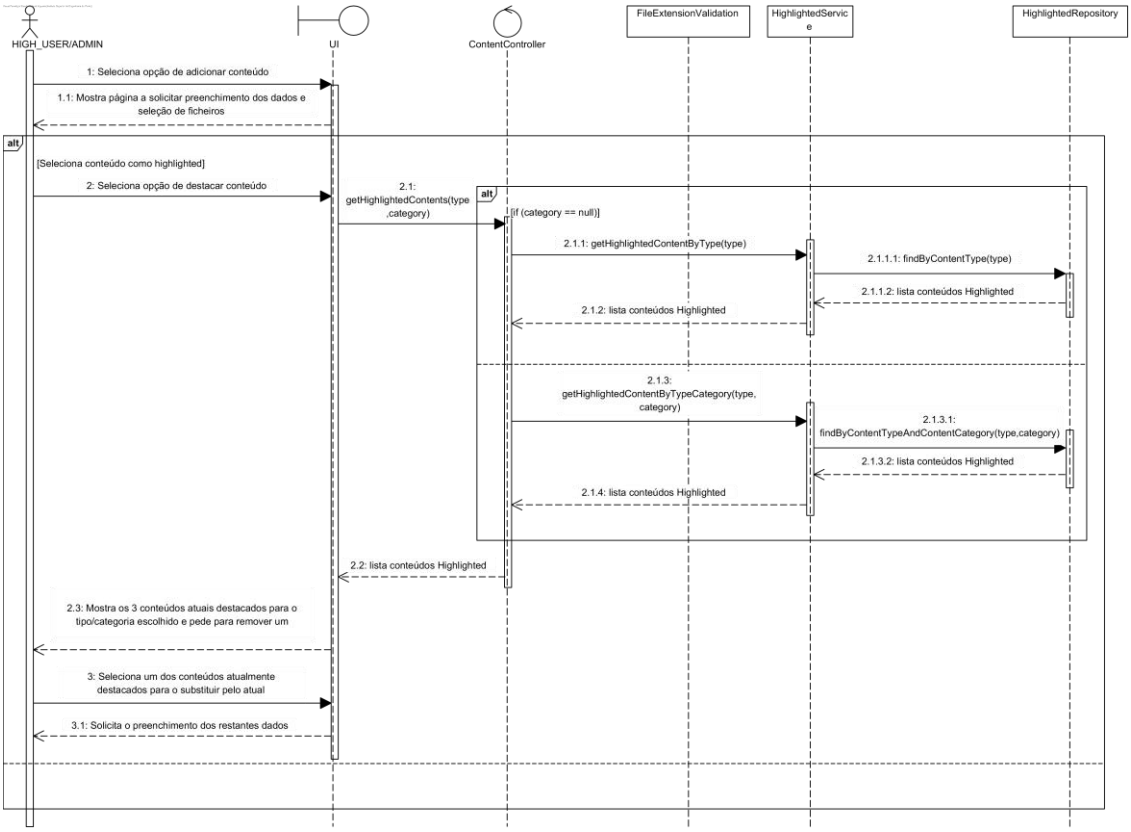



Figura 50 – Diagrama de sequência do caso de uso Criar Conteúdo

Após a seleção da opção de adicionar um novo conteúdo, disponível para os utilizadores com permissões de “High_User” e “Admin”, o sistema solicita o preenchimento de um formulário de dados. Nesse formulário, existe uma opção de seleção que permite colocar o conteúdo a ser criado em destaque. Caso o utilizador selecione essa opção, é espoletado um pedido à “LibraryAPI” para obter os conteúdos destacados para o tipo e/ou categoria selecionados previamente no formulário de dados. Em resposta a esse pedido, a API retorna então uma lista dos conteúdos destacados e, caso já existam três conteúdos atualmente em destaque, é solicitado ao utilizador para selecionar um desses conteúdos de forma a substituí-lo na zona de destaque pelo conteúdo a ser criado.

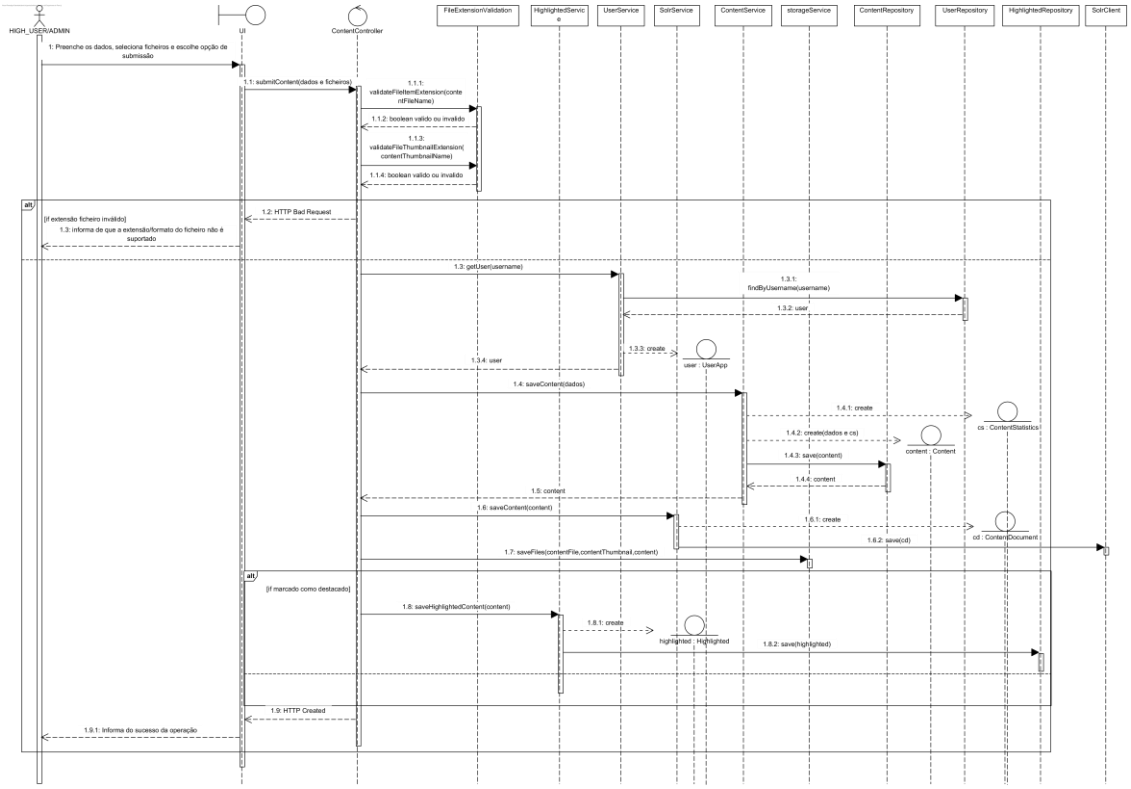



Figura 51 – Continuação diagrama de sequência do caso de uso Criar Conteúdo

Depois do preenchimento dos restantes dados obrigatórios e da seleção do ficheiro do conteúdo e da sua *thumbnail* (não obrigatório), o utilizador pode então submeter o conteúdo enviando todo o formulário e os ficheiros para a "LibraryAPI". Aqui, é efetuada uma validação dos ficheiros verificando se o seu formato é aceite ou não. Caso não seja, é enviada uma mensagem de erro para informar o utilizador. Caso seja, primeiramente é obtido o objeto do utilizador através de um pedido à base de dados, usando o *username* do utilizador autenticado no sistema. De seguida, é então iniciado o processo para armazenar o conteúdo na base de dados. Para isso, é criado um objeto de estatísticas do conteúdo ("ContentStatistics") para o associar ao objeto de conteúdo ("Content") a ser criado. Este objeto é populado com os dados do formulário e também com o nome do ficheiro do conteúdo e da sua *thumbnail*, sendo esse nome gerado pelo sistema por questões de segurança. Finalmente o conteúdo é armazenado na base de dados com os dados recebidos do formulário e a referência para as suas estatísticas e o utilizador que o criou.

Para além de ser guardado na base de dados, é também necessário armazenar alguns metadados do conteúdo no motor de pesquisa. No Extrato de código 7 é apresentado o método que tem essa responsabilidade.

```
@Override
    public void saveContent(Content content) throws IOException,
SolrServerException {
        ContentDocument doc = new
ContentDocument(content.getId(),content.getTitle(), content.getSummary(),
content.getUserCreated().getName(),
```

```
                content.getAuthors(), content.getFileItem(),
                content.getThumbnailItem(), content.getFormat(),
content.getKeywords(), content.getType(), content.getCategory(),
                content.getLanguage(), content.getUploadDate(),
content.getContentStatistics().getViewsCount(),
                content.getContentStatistics().getRating(),
content.getContentStatistics().getTotalRatingVotes());
        solrClient.addBean(SOLR_CORE_NAME, doc);
        solrClient.commit(SOLR_CORE_NAME);
    }
```

Extrato de código 7 – Criação do objeto correspondente ao documento a armazenar no Solr

No método acima visível, é criado um objeto de "ContentDocument" com os dados do conteúdo necessários e este é enviado para ser armazenado no Solr usando a API SolrJ. O "ContentDocument" é uma classe construída na "LibraryAPI" para representar a estrutura do documento a ser interpretado e armazenado pelo Solr, sendo a configuração dos seus atributos efetuada com recurso à própria notação da SolrJ.

Relativamente aos ficheiros do conteúdo submetido e da sua *thumbnail*, estes são recebidos no controlador como *MultipartFile* e são posteriormente tratados no serviço de armazenamento. No Extrato de código 8 é apresentada a implementação do método de armazenamento.

```
@Override
    public void saveFiles(MultipartFile fileItem, MultipartFile
fileThumbnail, Content content) throws IOException {
        if(fileItem != null) {
            new File(SAVE_FILE_DIRECTORY + content.getType().replaceAll("
", "_").toLowerCase()).mkdirs();
                try(InputStream is = fileItem.getInputStream()) {
                    Files.copy(is, Paths.get(SAVE_FILE_DIRECTORY +
content.getType().replaceAll(" ", "_").toLowerCase() + "\\" +
content.getFileItem()));
                }
        }
        if(fileThumbnail != null
&& !fileThumbnail.getOriginalFilename().equals(DEFAULT_THUMBNAIL)) {
            byte[] bytesThumbnailItem = fileThumbnail.getBytes();
            Path pathThumbnailItem = Paths.get(SAVE_THUMBNAIL_DIRECTORY +
content.getThumbnailItem());
            Files.write(pathThumbnailItem, bytesThumbnailItem);
       }
    }
```

Extrato de código 8 – Método de armazenamento do ficheiro do conteúdo e *thumbnail*

Neste método, é obtido o fluxo de *bytes* dos ficheiros de forma a proceder à sua gravação e armazenamento, sendo armazenados com os nomes gerados anteriormente pelo sistema. O armazenamento é efetuado no sistema de ficheiros do servidor numa localização definida que serve como repositório dos ficheiros. Esse repositório está ordenado pelas áreas de conteúdos existentes.

No caso de o utilizador ter marcado o conteúdo para destaque, é criado um objeto correspondente à entidade “Highlighted” e este é armazenado na base de dados com alguns dados e com a referência para o conteúdo criado.

Após a execução de todos os passos anteriores com sucesso, o utilizador é informado do sucesso da operação.

## 6.4 Visualizar Conteúdos

Nesta secção apresenta-se o diagrama de sequência e a descrição do processo de visualização de conteúdos. Este exemplo representa o caso em que se procede à visualização de conteúdos de um determinado tipo e categoria, mas é semelhante ao processo de visualização de conteúdos do utilizador, bem como da visualização de conteúdos de determinado tipo, mas sem categoria. Na Figura 52 é apresentado o diagrama de sequência deste processo.

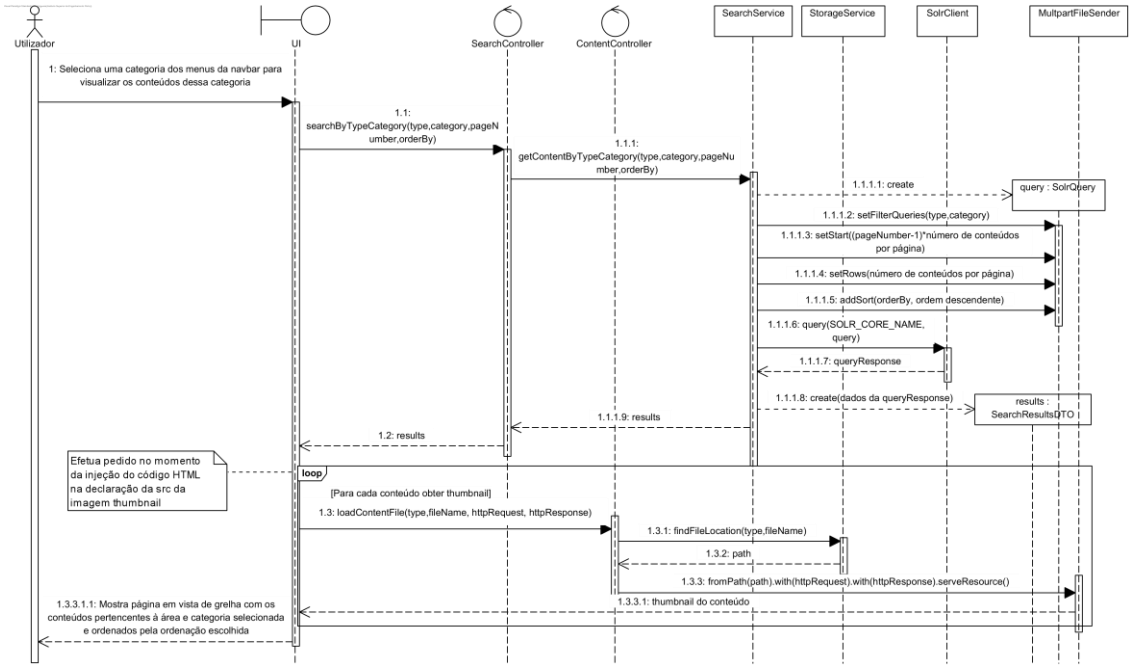



Figura 52 – Diagrama de sequênciado caso de uso Visualizar Conteúdos

Ao navegar na plataforma, qualquer utilizador pode selecionar uma categoria do menu de um tipo (área) de conteúdos disponível na barra de navegação. Ao efetuar essa ação, é enviado um pedido para a “LibraryAPI” para obter os conteúdos do tipo e categoria selecionados. Nesse pedido é recebido o tipo e categoria, o número da página a visualizar para ser usado na paginação e o filtro de ordenação. O controlador responsável por pedidos relacionados com o motor de pesquisa efetua uma chamada ao serviço de pesquisa que trata de instanciar um objeto do tipo “SolrQuery”, uma classe pertencente à API SolrJ. Neste objeto são definidas diversas propriedades e parâmetros que são interpretados pelo Solr como uma *query*. No

Extrato de código 9 é apresentada a implementação do método que trata de obter os conteúdos do tipo e categoria pretendidos.

```
@Override
public SearchResultsDTO getContentByTypeCategory(String type, String
category, int pageNumber, String orderBy) throws SolrServerException,
IOException {
        final SolrQuery query = new SolrQuery("*:*");
        query.setFilterQueries("type_s:\""+type+"\"",
"category_s:\""+category+"\"");
        query.setStart((pageNumber - 1) * NUMBER_CONTENTS_PAGE);
        query.setRows(NUMBER_CONTENTS_PAGE);
        query.addSort(orderBy, ORDER.desc);
        final QueryResponse response = solrClient.query(SOLR_CORE_NAME,
query);
        SearchResultsDTO searchResults = new
SearchResultsDTO(response.getBeans(ContentDocument.class),
response.getResults().getNumFound());
        return searchResults;
    }
```

Extrato de código 9 – Implementação do método para obter conteúdos por tipo e categoria

Assim, para obter os conteúdos pretendidos, neste caso define-se no objeto *query* o tipo e categoria de conteúdos a obter, a partir de que índice no Solr devem ser recolhidos os resultados e quantos resultados devem ser retornados para estar de acordo com a paginação solicitada pelo utilizador e, por fim, o filtro de ordenação a aplicar na *query* de pesquisa. Depois de configurado o objeto atrás referido, este é usado no pedido ao Solr através do "SolrClient" que devolve uma resposta com os resultados correspondentes à *query* executada. Estes resultados são tratados de forma a puderem ser interpretados pelo cliente através da criação de um objeto do tipo "SearchResultsDTO" que é enviado para o cliente. Na construção do código HTML a injetar na parte do cliente, de forma a obter a *thumbnail* de cada conteúdo a apresentar, é efetuado um pedido à "LibraryAPI" que, para cada conteúdo, trata de obter o caminho do ficheiro correspondente à *thumbnail* e de retornar essa *thumbnail* para o cliente, mostrando assim os conteúdos da respetiva área e categoria selecionadas ao utilizador. Na Figura 53 é apresentado o resultado deste processo para o exemplo de conteúdos pertencentes ao tipo *Software Development* e categoria HTML5.

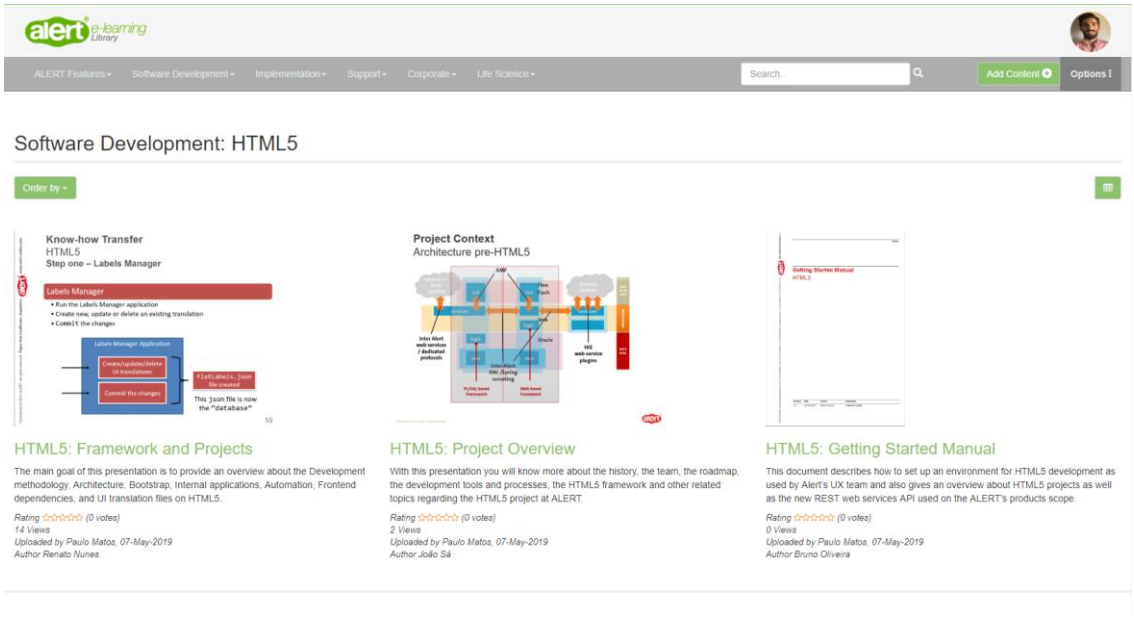



Figura 53 – Vista de conteúdos

## 6.5 Visualizar Detalhes do Conteúdo

Na Figura 54 é apresentado o diagrama de sequência relativo ao requisito de visualizar detalhes de um conteúdo e é efetuada uma descrição do mesmo.

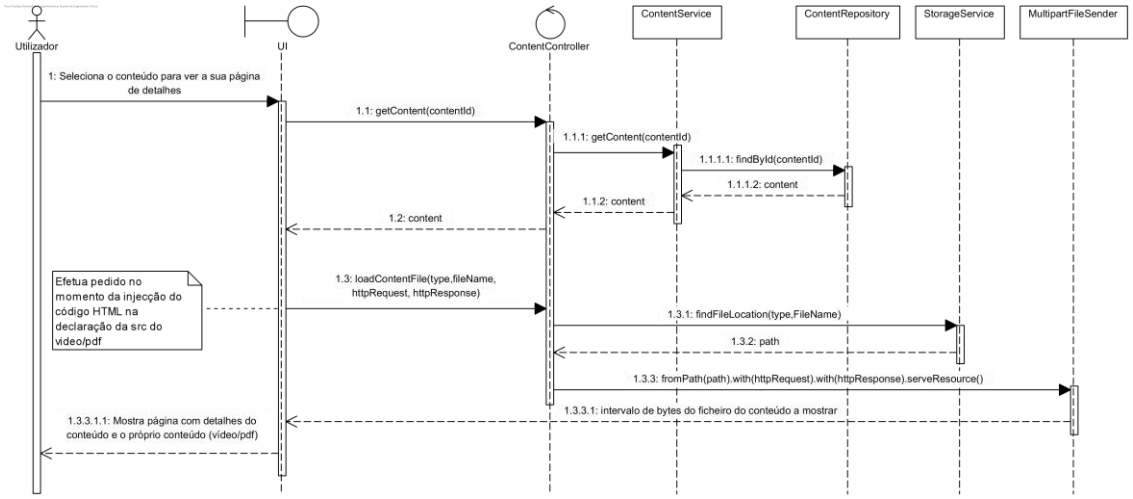



Figura 54 –Diagrama de sequência do caso de uso de Visualizar Detalhes do Conteúdo

Ao selecionar um conteúdo para visualizar os seus detalhes, é efetuado um pedido à "LibraryAPI" intercetado pelo controlador responsável pelos pedidos relacionados com conteúdos. O controlador recebe o id do conteúdo a visualizar e invoca o método do serviço de conteúdos. A implementação deste método é apresentada no Extrato de código 10.

```
@Override
    public Content getContent(long contentId) {
```

```
        return contentRepo.findById(contentId).orElse(null);
    }
```

Extrato de código 10 – Implementação do método para obter um conteúdo do repositório pelo seu id

Este método pede ao repositório de conteúdos para obter o conteúdo da base de dados cujo id corresponde ao id recebido, resultando numa instância do Conteúdo que é então retornado para o cliente com todos os detalhes do conteúdo. Caso não exista um conteúdo com o id solicitado é retornado null. Durante a construção do HTML a injetar na parte do cliente, para apresentar o conteúdo do ficheiro de vídeo ou PDF, é efetuado um pedido à "LibraryAPI" que trata de obter o caminho desse ficheiro e, através do caminho, obter os *bytes* do ficheiro e retorná-los para o cliente, mostrando assim o conteúdo do ficheiro. Na Figura 55 é apresentada parte da página de visualização de detalhes de um conteúdo, resultado do processo descrito.

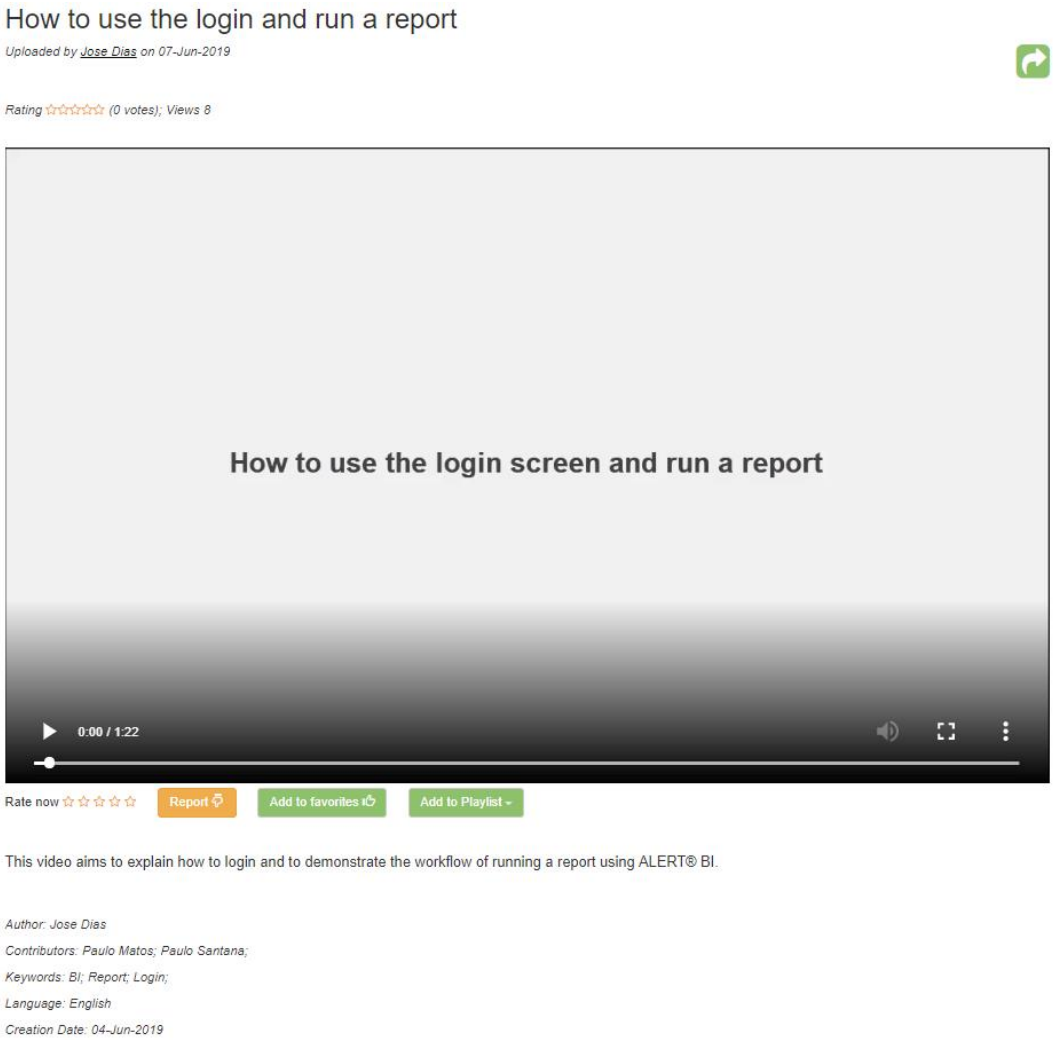



Figura 55 - Visualização de detalhes do conteúdo

Caso o utilizador autenticado fosse o utilizador que submeteu o conteúdo ou um administrador, iriam estar presentes as opções de editar e remover o conteúdo visualizado ao lado da opção de o adicionar a uma *Playlist,* visível na figura anterior (Figura 55).

## 6.6 Pesquisar por Conteúdo

Na Figura 56 é apresentado o diagrama de sequência do requisito de pesquisar por conteúdo e uma descrição do mesmo.

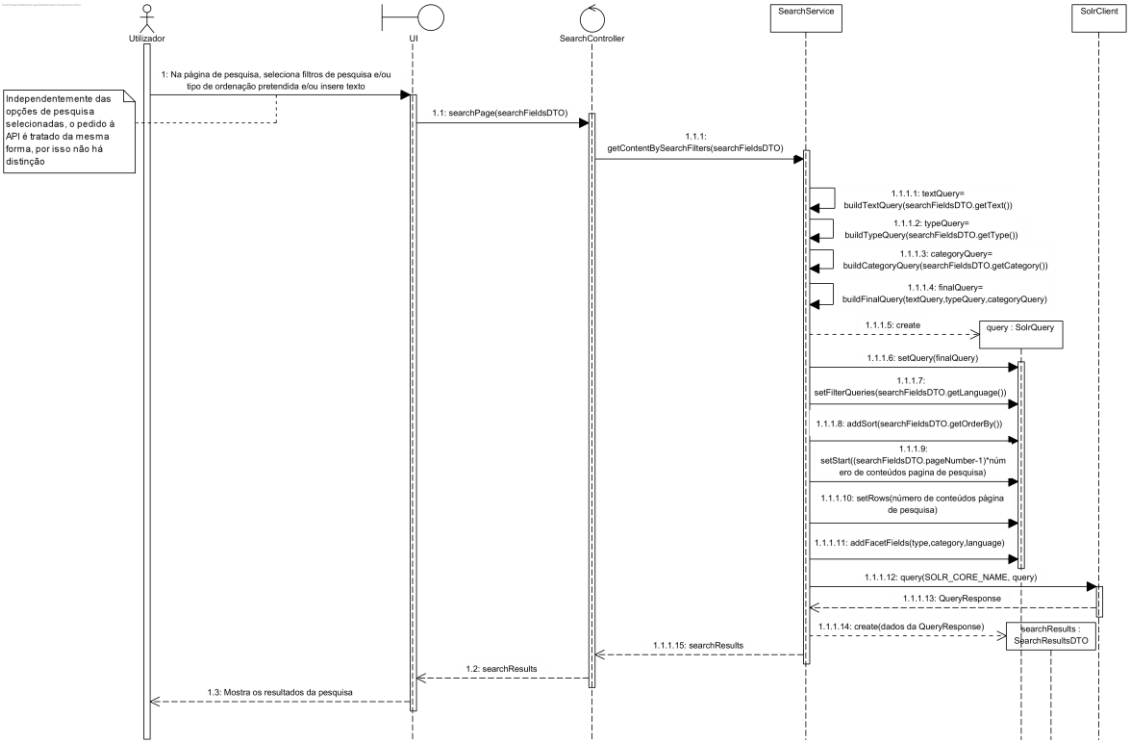



Figura 56 – Diagrama de sequência do caso de uso Pesquisar por Conteúdo

Quando o utilizador acede à página de pesquisa sem ter inserido nenhum texto no campo de pesquisa da barra de navegação, são mostrados por predefinição os últimos conteúdos adicionados à plataforma, podendo o utilizador refinar a pesquisa como desejar. Parte dessa página de pesquisa é apresentada na Figura 57.

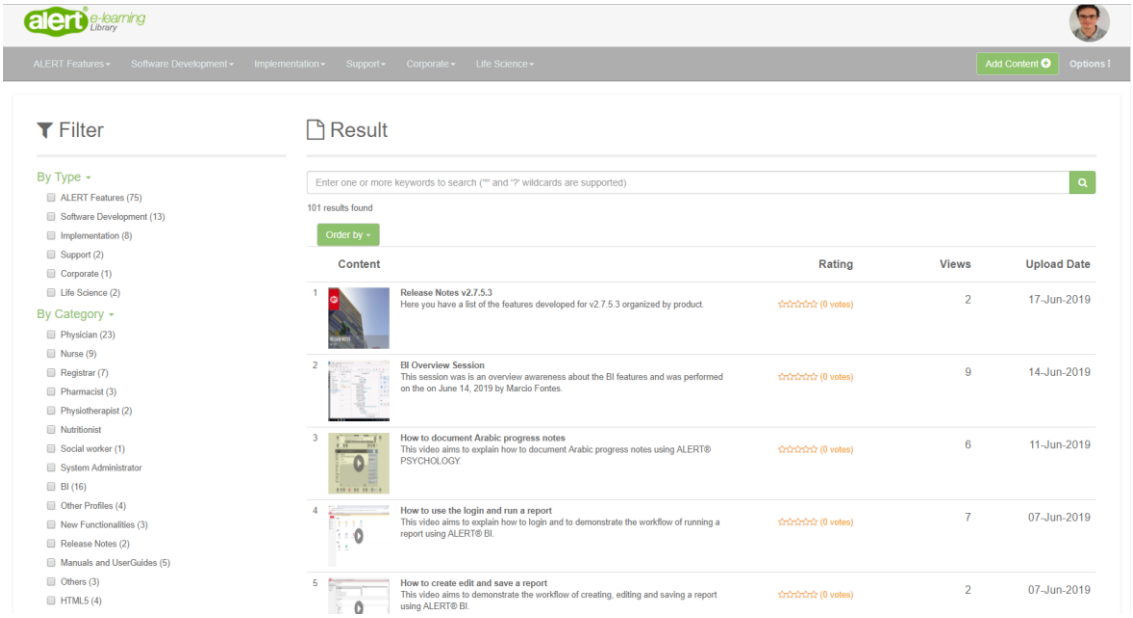



Figura 57 – Página de pesquisa de conteúdos

Para procurar por conteúdos, o utilizador pode selecionar filtros de pesquisa que consistem no tipo, categoria e língua do conteúdo. Todos estes filtros são de pesquisa facetada. Para além dos filtros, o utilizador pode inserir texto associado à pesquisa e também ordenar os resultados obtidos por data, classificação ou número de visualizações, podendo também percorrer as diferentes páginas de resultados através da paginação. Todos estes dados da pesquisa inseridos pelo utilizador são enviados para a "LibraryAPI" onde são interpretados pelo controlador como um objeto do tipo "SearchFieldsDTO". No Extrato de código 11 é apresentado o método do controlador que recebe o pedido de pesquisa.

```
@PostMapping("")
    @ResponseBody
    public ResponseEntity<SearchResultsDTO> searchPage(@RequestBody
SearchFieldsDTO searchFields)
    {
        SearchResultsDTO searchResult = new SearchResultsDTO();
        try {
            searchResult =
solrService.getContentBySearchFilters(searchFields);
            return ResponseEntity.ok(searchResult);
        } catch (SolrServerException | IOException e) {
            e.printStackTrace();
            return new ResponseEntity<>(HttpStatus.INTERNAL_SERVER_ERROR);
        }
    }
```

Extrato de código 11 – Método do controlador de pesquisa que recebe o pedido de pesquisa

O controlador chama então um método do serviço de pesquisa passando o objeto recebido com as opções de pesquisa. Nesse método, são criadas as *queries* para pesquisar pelo texto inserido pelo utilizador em diferentes campos de metadados do conteúdo e para pesquisar pelo tipo e categoria do conteúdo, convergendo todas elas numa *query* final. De seguida, é criada uma instância do objeto do tipo "SolrQuery" onde é incluída a *query* anteriormente obtida e são

definidos os restantes parâmetros para a pesquisa, como a língua do conteúdo, o filtro de ordenação, o índice no Solr onde deve ser iniciada a recolha de resultados e o número de resultados a obter. Para além disso, são adicionados os campos de pesquisa facetada para informar o utilizador do número de resultados encontrados para cada faceta. Definido o objeto "SolrQuery", é então enviado o pedido ao Solr através do "SolrClient" e são obtidos os resultados da resposta à *query* executada. De forma a enviar os resultados para o cliente e estes serem interpretados com maior facilidade, é criado um objeto do tipo "SearchResultsDTO" que é populado com os resultados tratados e é então enviado para o cliente, onde são mostrados os resultados da pesquisa ao utilizador. Para apresentar o número de conteúdos encontrados para os filtros que fazem parte da pesquisa facetada, é necessário tratar os dados recebidos do motor de pesquisa. No Extrato de código 12 é apresentado o método que efetua esse tratamento.

```
public Map<String,Integer> readFacets(List<FacetField> facetFields){
        Map<String,Integer> facets = new HashMap<String,Integer>();
        for(FacetField facetField : facetFields) {
            for(Count ff : facetField.getValues()) {
                facets.put(ff.getName(), (int) ff.getCount());
            }
        }
        return facets;
    }
```

Extrato de código 12 – Método para tratar os resultados da pesquisa por facetas

Neste método é recebida uma lista de objetos do tipo "FacetField" que contém, entre outros dados, o campo da faceta e o número de resultados encontrados para essa faceta. Percorrendo essa lista, para cada objeto FacetField é adicionado a um Map o nome do campo da faceta e o número de conteúdos encontrados pertencentes a esse campo. Este Map é então retornado e adicionado ao objeto "SearchResultsDTO".

## 6.7 Adicionar Conteúdo aos seus Favoritos

Na Figura 58 é possível visualizar o diagrama de sequência da funcionalidade de adicionar um conteúdo aos favoritos seguido da sua descrição.

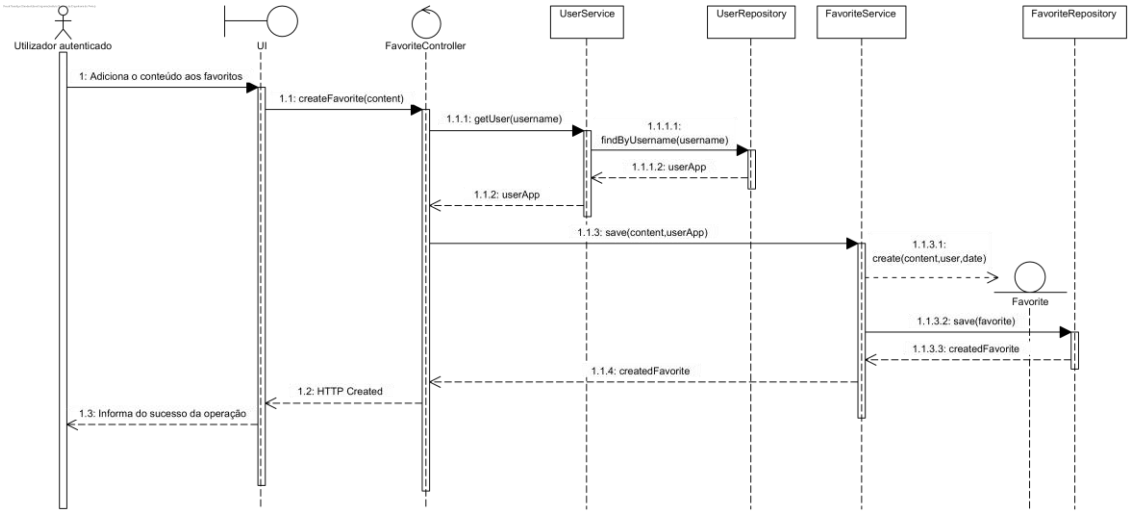



Figura 58 – Diagrama de sequência do caso de uso Adicionar Conteúdo aos seus Favoritos

Um conteúdo pode ser adicionado aos favoritos do utilizador autenticado através da seleção dessa opção na página de detalhes de um conteúdo (visível na Figura 55). Após a seleção dessa opção, é espoletado o envio de um pedido para a "LibraryAPI" com o conteúdo a ser adicionado aos favoritos do utilizador. O controlador de favoritos obtém o objeto do utilizador através de um pedido à base de dados intermediado pelo serviço e repositório de utilizadores, usando o *username* do utilizador autenticado no sistema. De seguida, o serviço dos favoritos cria uma instância do objeto de favoritos com o conteúdo, o utilizador e a data em que o favorito foi adicionado e invoca o repositório de favoritos para armazenar esse objeto na base de dados. Por fim, a "LibraryAPI" retorna o código "HTTP Created" que indica que a requisição foi bem-sucedida e o novo recurso criado, sendo assim o utilizador informado do sucesso da operação. No Extrato de código 13 é apresentado o método do controlador de favoritos referido que recebe o pedido e envia a resposta para o cliente.

```
@PostMapping("")
    public ResponseEntity<Favorite> createFavorite(@RequestBody Content
content) {
        UserApp user =
userService.getUser(SecurityContextHolder.getContext().getAuthentication().
getName());
        return ResponseEntity.ok(favoriteService.save(content, user));
    }
```

Extrato de código 13 – Método do controlador de favoritos que recebe o pedido de adição de favorito

## 6.8 Visualizar Favoritos

Nesta secção é apresentado o processo de visualização de conteúdos favoritos do utilizador através da Figura 59 e da sua descrição. O processo de visualização do histórico do utilizador é semelhante ao a seguir apresentado, mudando apenas as classes e entidades responsáveis por obter o histórico de conteúdos visualizados. Este processo é também semelhante ao de visualização de conteúdos descrito anteriormente (secção 6.4), mas difere na fonte de obtenção

dos resultados, uma vez que neste caso estes são obtidos da base de dados e não do motor de pesquisa.

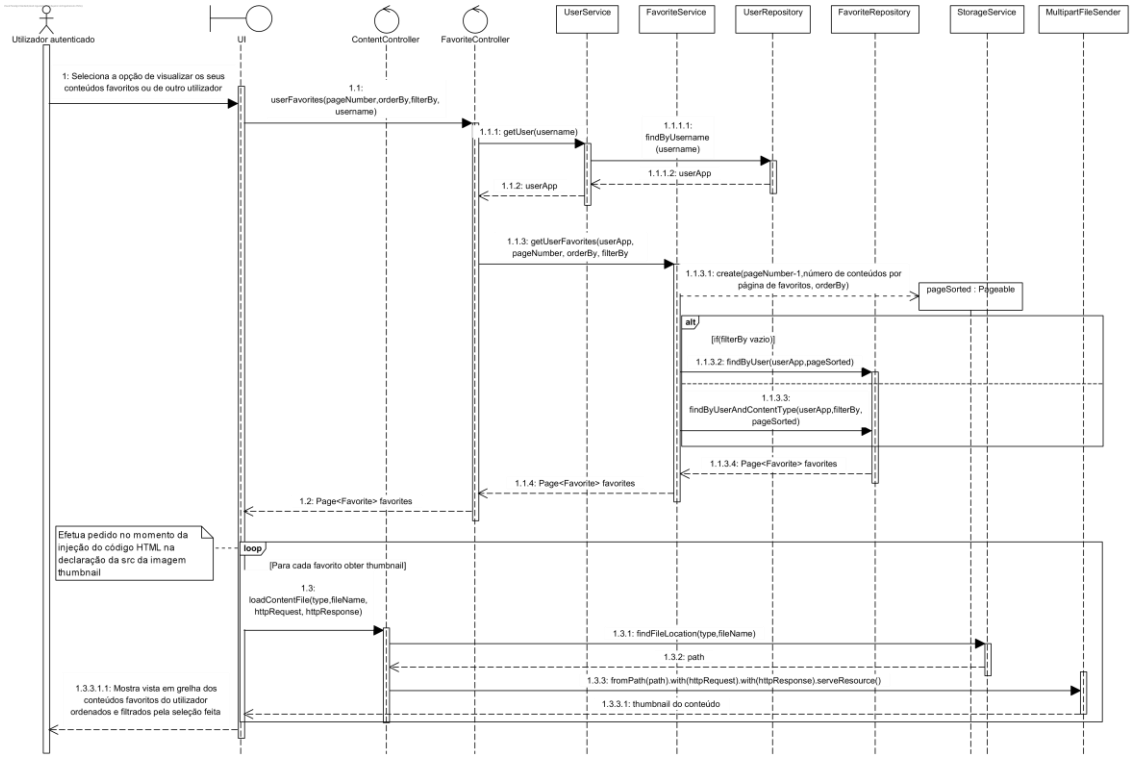



Figura 59 – Diagrama de sequência do caso de uso Visualizar Favoritos

O utilizador autenticado pode visualizar os seus conteúdos favoritos ou os favoritos de outro utilizador. Após selecionar a opção de visualizar favoritos, é espoletado um pedido à "LibraryAPI" com o número da página atual de visualização para paginação, filtros de ordenação e o *username* cujos favoritos pretende consultar (o próprio utilizador ou outro). Quando se entra pela primeira vez na página de consulta de favoritos, estes valores assumem valores predefinidos (por exemplo, o número da página assume o valor 1). Recebido o pedido no controlador de favoritos da "LibraryAPI", é obtido o objeto do utilizador do qual se quer consultar os favoritos através de um pedido à base de dados, efetuado pelo repositório de utilizadores. O passo seguinte consiste em invocar um método do serviço de favoritos, onde é criado um objeto do tipo "Pageable" existente no Spring Data. Este objeto permite definir configurações para obter os favoritos da base de dados com o conceito de paginação e ordenação. No Extrato de código 14 é apresentado o método do serviço de favoritos referido.

```
@Override
    public Page<Favorite> getUserFavorites(UserApp user, int pageNumber,
String orderBy, String filterBy){
        Pageable pageSorted = null;
        if(orderBy.equalsIgnoreCase("dateAdded")) {
            pageSorted = PageRequest.of(pageNumber-1, NUMBER_CONTENTS_PAGE,
Sort.by(orderBy).descending());
        } else if(orderBy.equalsIgnoreCase("viewsCount")) {
            pageSorted = PageRequest.of(pageNumber-1, NUMBER_CONTENTS_PAGE,
Sort.by("Content.ContentStatistics.viewsCount").descending());
```

```
        } else if (orderBy.equalsIgnoreCase("rating")) {
            pageSorted = PageRequest.of(pageNumber-1, NUMBER_CONTENTS_PAGE,
Sort.by("Content.ContentStatistics.rating").descending());
        }
        if(filterBy.isEmpty() || filterBy.equals("favoriteAllTypes")) {
            return favoriteRepo.findByUser(user, pageSorted);
        } else {
            return favoriteRepo.findByUserAndContentType(user, filterBy,
pageSorted);
        }
    }
```

Extrato de código 14 – Implementação do método de obtenção de favoritos do utilizador no serviço de favoritos

Neste caso, o objeto “Pageable” é criado com o número da página a consultar, o número de conteúdos incluídos em cada página de favoritos, ou seja, o número de conteúdos a obter da base de dados, e o filtro de ordenação selecionado. De seguida é feita a chamada à base de dados por parte do repositório de favoritos para obter os favoritos do utilizador, tendo em conta as configurações presentes no objeto “Pageable”. Caso o utilizador tenha selecionado a opção de filtragem por tipo de conteúdo, o pedido à base de dados é efetuado tendo também em conta o filtro escolhido, obtendo assim apenas os favoritos com conteúdos pertencentes a esse tipo. Os resultados obtidos são retornados num objeto do tipo “Page”, também disponibilizado no Spring Data, que contém a lista de favoritos obtidos da base de dados, bem como informações adicionais úteis para a paginação de resultados, como o número total de elementos, o número total de páginas, entre outras. Na construção do código HTML na parte do cliente, para apresentar a *thumbnail* de cada conteúdo favorito a apresentar, é efetuado um pedido à “LibraryAPI” que trata de obter o caminho do ficheiro correspondente à *thumbnail* e de retornar essa *thumbnail* para o cliente, mostrando assim os favoritos do utilizador. Na Figura 60 é visível parte da página de favoritos do utilizador autenticado com os seus conteúdos favoritos e as opções de ordenação e filtração.

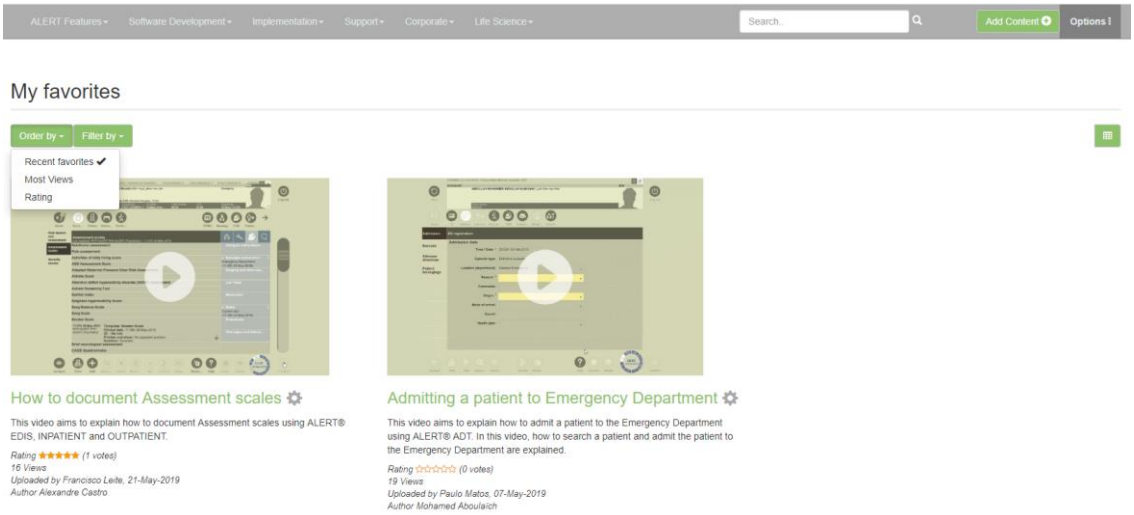



Figura 60 - Vista de conteúdos favoritos do utilizador

Cada favorito apresenta um menu de opções, disponível ao lado do título de cada conteúdo através do símbolo da roda dentada, visível na figura acima, onde o utilizador pode selecionar a opção de visualizar os detalhes desse conteúdo favorito ou de o remover dos seus favoritos. A remoção consiste numa operação típica à base de dados em que o favorito é removido através do seu id.

## 6.9 Criar *Playlist*

Na Figura 61 é apresentado um diagrama de sequência representativo da criação de uma *Playlist* seguido de uma descrição do mesmo.

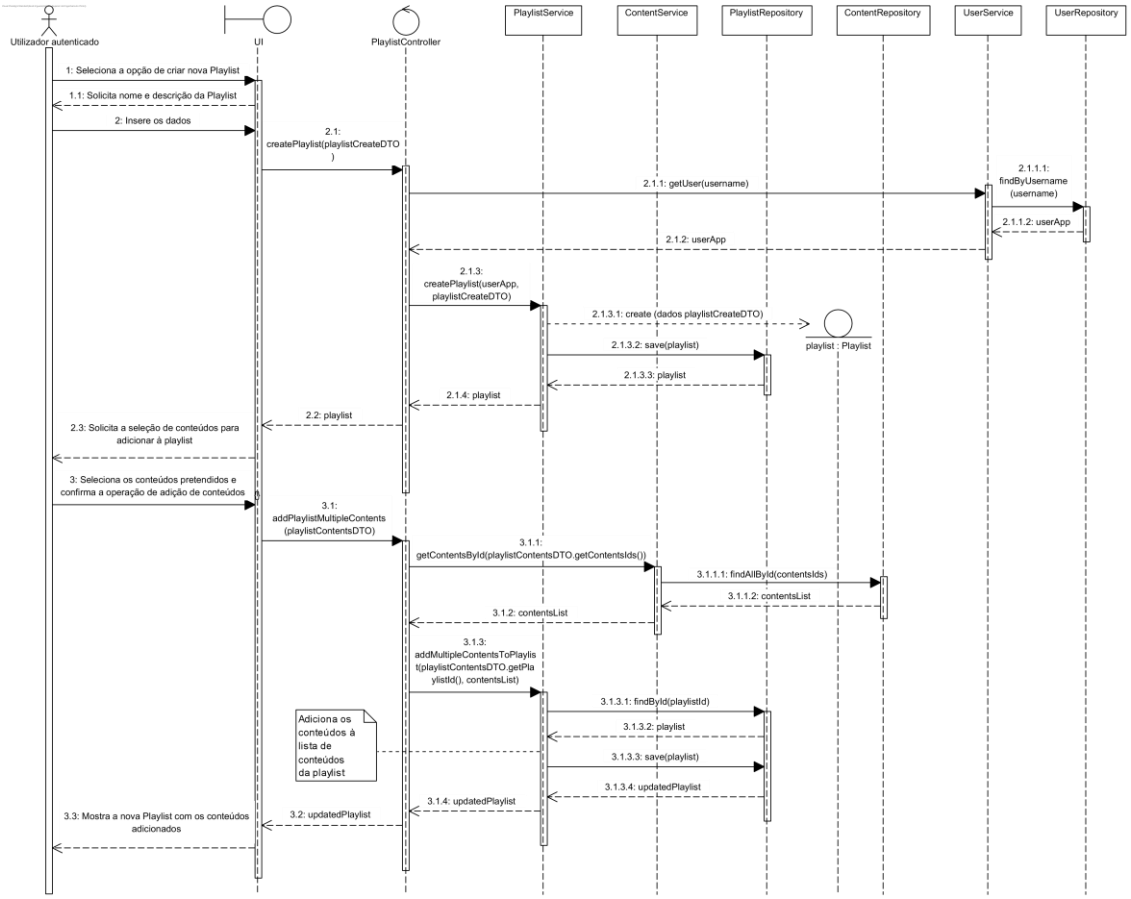



Figura 61 – Diagrama de sequência do caso de uso Criar *Playlist*

O processo é iniciado pela seleção da opção de criação de uma nova *Playlist* na área de visualização de *Playlists*, sendo solicitado um nome e uma descrição da *Playlist* a ser criada. Após a inserção e submissão dos dados, estes são enviados para a "LibraryAPI" com um formato correspondente a um objeto do tipo "PlaylistCreateDTO". De seguida, é efetuado um pedido à base de dados intermediado pelo serviço e repositório de utilizadores para obter o objeto do utilizador autenticado. Com estes dois objetos, é então invocado o método do serviço de *Playlists* para criar um objeto do tipo "Playlist" com o nome e descrição presentes no objeto

recebido e o utilizador criador da mesma. Depois de armazenada a *Playlist* na base de dados, é solicitado ao utilizador para selecionar os conteúdos que pretende adicionar à *Playlist* criada. O armazenamento da *Playlist* é efetuado antes da adição de conteúdos, uma vez que existe outro processo diferente de criação de *Playlists* que tem em comum a primeira parte descrita da criação e armazenamento da *Playlist*. Após essa seleção e submissão por parte do utilizador, o repositório de conteúdos obtém da base de dados uma lista com todos os objetos correspondentes aos conteúdos selecionados. Obtida essa lista, é chamado um método do serviço de *Playlists* que tem a responsabilidade de adicionar os conteúdos dessa lista à *Playlist*. No Extrato de código 15 é apresentada a implementação desse método.

```
@Override
    public Playlist addMultipleContentsToPlaylist(long playlistId,
List<Content> contents) {
        Playlist playlist = playlistRepo.findById(playlistId).orElse(null);
        for(Content content: contents) {
            if(!playlist.getContents().contains(content)) {
                playlist.getContents().add(content);
            }
        }
        return playlistRepo.save(playlist);
    }
```

Extrato de código 15 – Implementação do método para atualizar lista de conteúdos de uma *Playlist*

Neste método, é obtida uma instância da *Playlist* através de um pedido ao repositório usando o id da *Playlist* recebido por parâmetro. Tendo o objeto da *Playlist*, é percorrida a lista de conteúdos selecionados pelo utilizador e, para cada conteúdo nessa lista, é verificado se a *Playlist* atual já contém ou não esse conteúdo para evitar duplicações. Caso não contenha, o conteúdo é adicionado à lista de conteúdos atual da *Playlist*. Este método é utilizado tanto para a adição de conteúdos a uma *Playlist* já existente com conteúdos como para uma *Playlist* acabada de criar. Neste caso, como a *Playlist* acabou de ser criada, todos os conteúdos selecionados pelo utilizador vão ser adicionados à *Playlist*. Tendo a lista de conteúdos da *Playlist* atualizada, o seu objeto é passado para o repositório onde é efetuada essa atualização na base de dados. A *Playlist* atualizada é então retornada para o controlador e posteriormente para o cliente, onde pode ser visualizada com os novos conteúdos já adicionados. Na Figura 62 é apresentada a vista da página de *Playlists*, sendo possível navegar entre as *Playlists* existentes e entre os seus conteúdos.

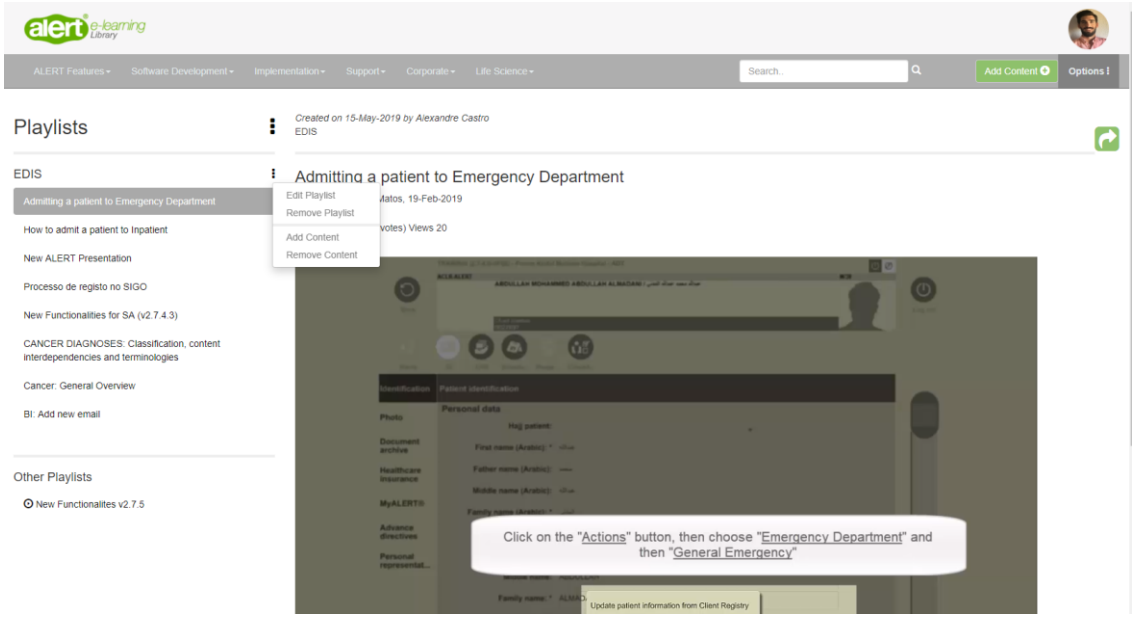



Figura 62 - Página de visualização de *Playlists*

## 6.10 Partilhar Cópia da *Playlist*

Nesta secção é apresentado o diagrama de sequência e uma descrição do caso de uso de partilhar uma *Playlist*. Na plataforma, para além de ser possível partilhar conteúdos, é também possível partilhar uma *Playlist* de três formas distintas: diretamente na plataforma com outro utilizador, por email ou por WhatsApp. Na Figura 63 é apresentado esse menu de opções de partilha, existindo também a possibilidade de copiar o URL que permite visualizar a *Playlist* aberta.

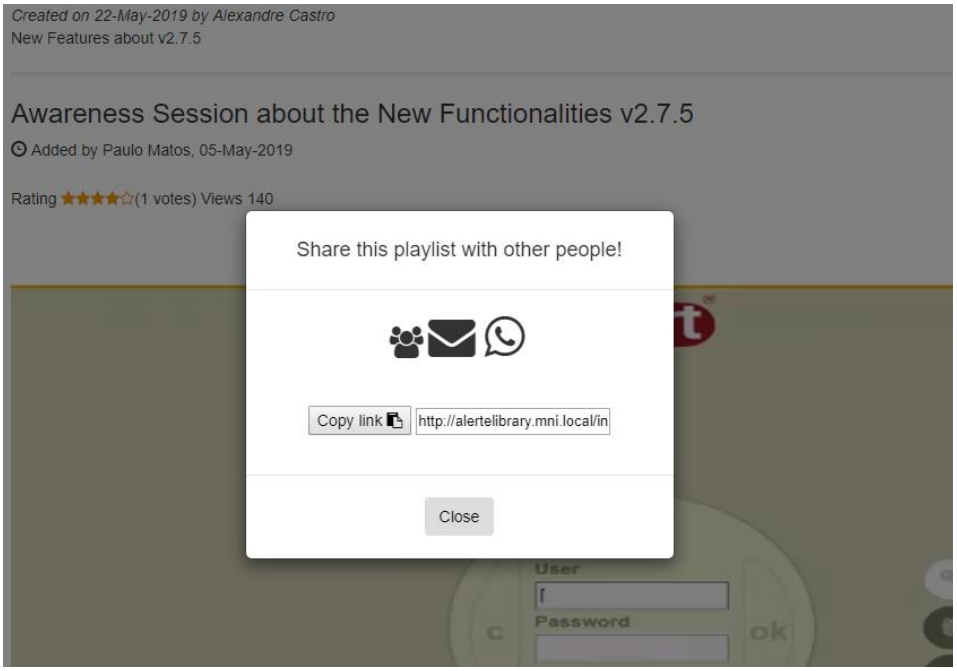



Figura 63 – Opções de partilha de *Playlist*

O processo que se segue (Figura 64) refere-se à primeira forma de partilha, a partilha de uma *Playlist* com outros utilizadores na plataforma. Os outros dois meios de partilha são de execução direta no cliente sem ser necessário comunicar com a LibraryAPI.

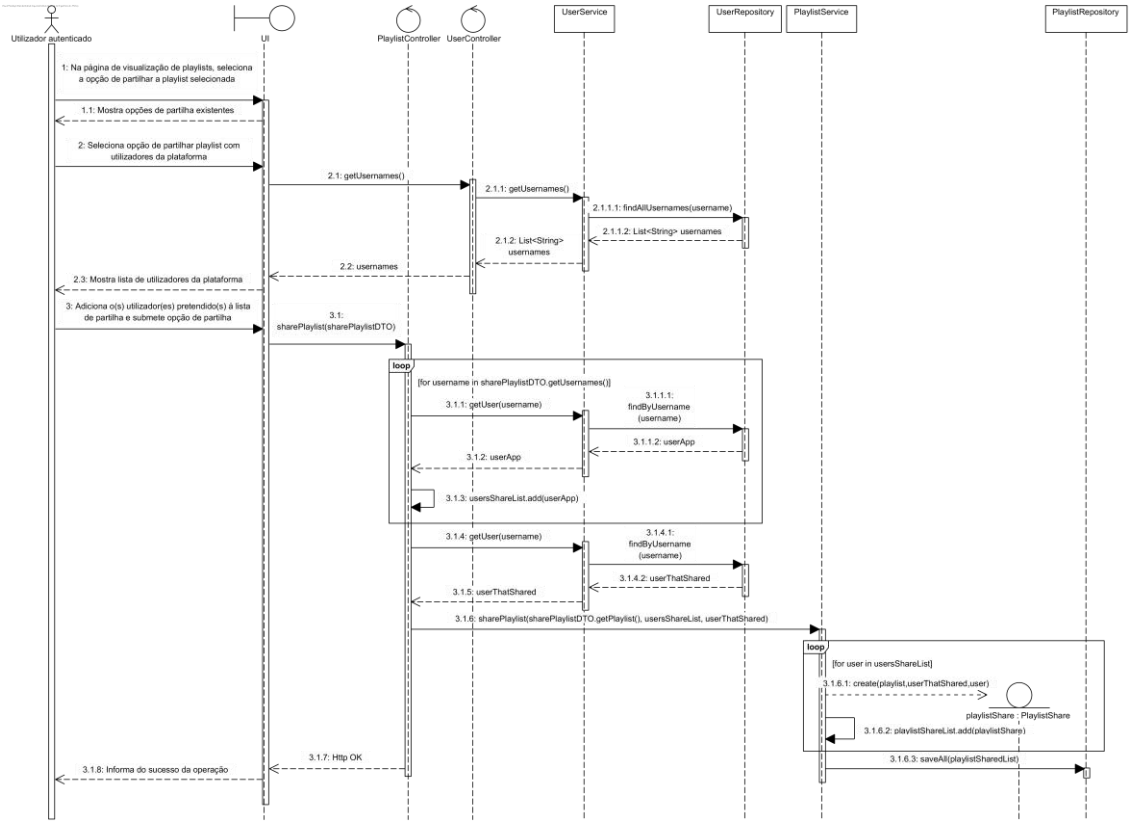



Figura 64 – Diagrama de sequência do caso de uso Partilhar Cópia da *Playlist*

O processo de partilha de uma *Playlist* pode ser efetuado por qualquer utilizador registado e é iniciado pela seleção da opção de partilha na página de visualização de *Playlists*. O sistema mostra as opções de partilha anteriormente referidas e, ao ser escolhida a opção de partilhar a *Playlist* com outro utilizador da plataforma, é enviado um pedido à "LibraryAPI" para obter os *usernames* dos utilizadores registados no sistema. Assim, o repositório de utilizadores efetua uma *query* à base de dados para obter os *usernames* de todos os utilizadores por ordem alfabética, exceto o do utilizador atualmente autenticado, uma vez que este é o utilizador que pretende partilhar a *Playlist*. Essa *query* é exposta no Extrato de código 16.

```
@Query("select username from UserApp where not username = ?1 order by
username")
    List<String> findAllUsernames(String username);
```

Extrato de código 16 – *Query* à base de dados para obter todos os usernames exceto o passado por parâmetro

Com a lista de *usernames* obtida, são então apresentados os *usernames* dos utilizadores registados na plataforma para que o utilizador possa selecionar aqueles com que pretende partilhar a *Playlist*. A vista desta opção de seleção dos utilizadores é apresentada na Figura 65.

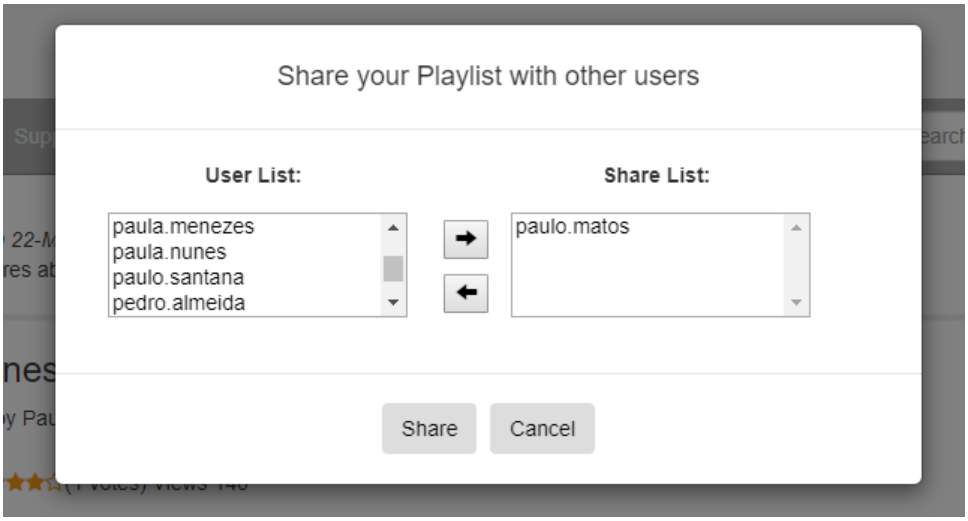



Figura 65 – Seleção de utilizadores para partilha da *Playlist*

Após a seleção dos utilizadores pretendidos e da submissão da partilha, o controlador responsável por operações relacionadas com *Playlists* recebe os dados num objeto do tipo "SharePlaylistDTO". Neste objeto está presente a lista de *usernames* selecionados pelo utilizador para partilhar a *Playlist* e, para cada *username* presente nessa lista, é obtida da base de dados e armazenada numa lista uma instância do utilizador correspondente a esse *username*. Para além desses objetos relativos aos utilizadores com quem se pretende partilhar a *Playlist*, é também criado um objeto relativo ao utilizador que está a partilhar a *Playlist*. Por fim, é invocado um método do serviço de *Playlists* que, para cada utilizador presente na lista de utilizadores criada anteriormente, instância um objeto do tipo "Playlistshare" com a *Playlist* a partilhar, o utilizador que a partilhou e o utilizador alvo da partilha. Esse método é apresentado no Extrato de código 17.

```
@Override
    public void sharePlaylist(Playlist sharePlaylist, List<UserApp> users,
UserApp userShared) {
        List<Playlistshare> playlistshare = new ArrayList<>();
        for(UserApp user: users) {
            playlistshare.add(new Playlistshare(sharePlaylist, user,
userShared));
        }
        playlistshareRepo.saveAll(Playlistshare);
    }
```

Extrato de código 17 – Método para partilha da *Playlist* com os utilizadores selecionados

Depois de ter todos estes objetos de partilha criados numa lista, estes são armazenados na base de dados através do repositório de *Playlists* e o utilizador é informado do sucesso da operação.

## 6.11 Denunciar Conteúdo

Nesta secção é apresentado na Figura 66 o diagrama de sequência do processo de denúncia de conteúdos e uma respetiva descrição desse processo.

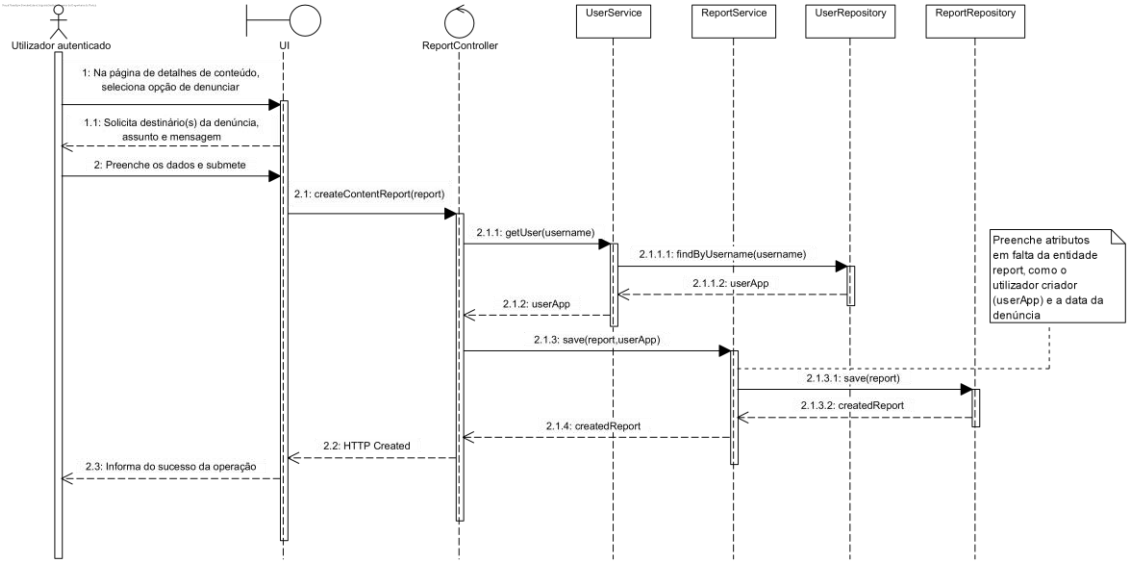



Figura 66 – Diagrama de sequência do caso de uso Denunciar Conteúdo

O processo de denúncia de um conteúdo é iniciado através da seleção da opção de denunciar conteúdo na página de detalhes desse conteúdo (visível na Figura 55). Ao selecionar essa opção, é solicitado ao utilizador autenticado para indicar o destinatário da denúncia, que pode ser o utilizador que adicionou o conteúdo à plataforma e/ou os administradores, bem como um assunto e uma mensagem. Na Figura 67 é apresentada a janela que é mostrada ao utilizador quando este pretende registar uma denúncia de um conteúdo.

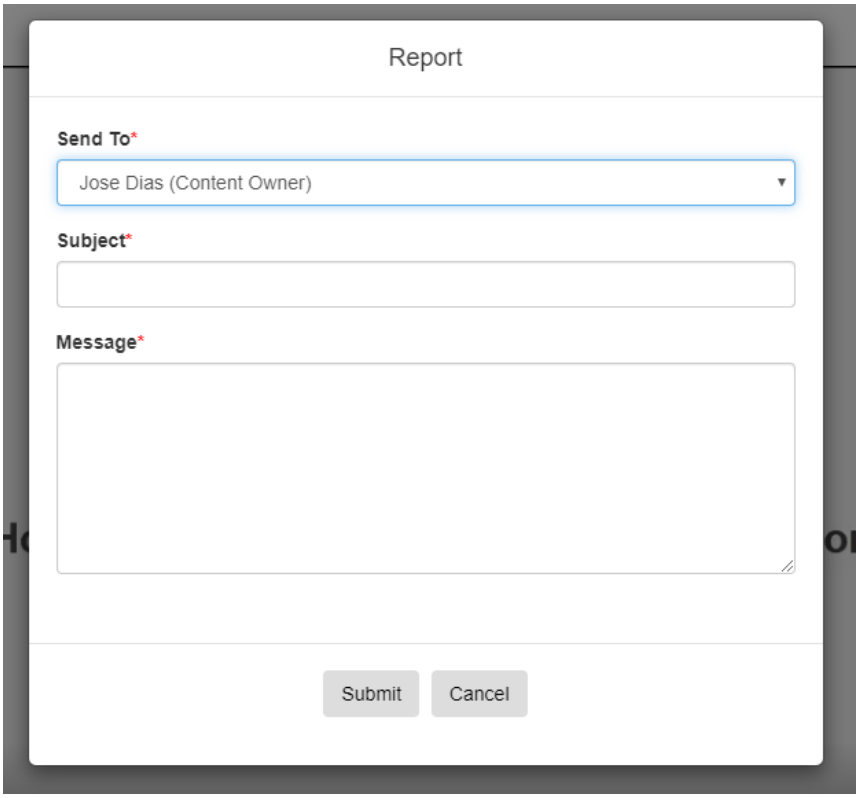



Figura 67 – Janela para a criação de uma denúncia

Inseridos os dados e confirmada a submissão da denúncia, é enviado um pedido para a "LibraryAPI" para a criação da denúncia. Recebido o pedido no controlador que tem a responsabilidade de tratar dos pedidos relativos às denúncias, é gerado o objeto correspondente ao utilizador criador da denúncia através de um pedido ao serviço de utilizadores que, consequentemente, chama o repositório de utilizadores, indicando o *username* do utilizador autenticado no sistema para o usar no pedido à base de dados. Com o objeto do utilizador, o controlador passa a denúncia recebida e esse utilizador para o serviço de denúncias, onde são inseridos os restantes atributos em falta no objeto da denúncia ("report"). Com todos os campos do objeto da denúncia populados, este é armazenado na base de dados através do repositório de denúncias, terminando o processo e avisando o utilizador do sucesso da operação. No Extrato de código 18 é apresentado o método do controlador de denúncias que trata do pedido de criação de uma denúncia.

```
@PostMapping("")
    public ResponseEntity<Report> createContentReport(@RequestBody Report
report) {
        UserApp user =
userService.getUser(SecurityContextHolder.getContext().getAuthentication().
getName());
        return ResponseEntity.ok(reportService.save(report, user));
    }
```

Extrato de código 18 – Método do controlador de denúncias que trata o pedido de registo de denúncia

# 7 Testes

Seguindo as boas práticas de desenvolvimento de *software*, ao longo da implementação do sistema foram realizados diversos testes, adotando uma abordagem iterativa que permitiu encontrar defeitos prematuramente e assegurar o correto funcionamento das funcionalidades desenvolvidas. Assim, efetuaram-se testes às principais funcionalidades do sistema, nomeadamente testes unitários às diferentes camadas da API desenvolvida (controladores, serviços, modelos e repositórios), bem como testes de integração para verificar se os componentes do sistema funcionam conforme esperado quando integrados. Apesar de não estarem demonstrados neste documento, foram realizados testes de aceitação juntamente com o supervisor do estágio de forma a validar se a implementação cumpre os requisitos. Na Figura 68 é apresentada a janela gerada com os resultados dos testes quando estes são executados.

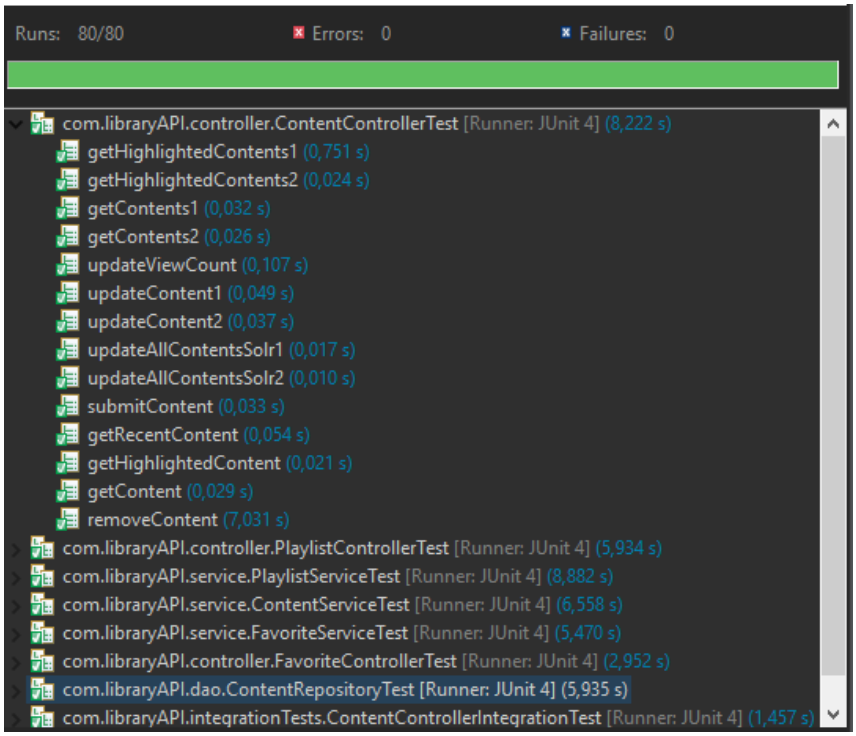



Figura 68 – Resultados dos testes realizados

No desenvolvimento dos testes unitários e dos testes de integração foi importado o módulo spring-boot-starter-test disponibilizado pela Spring Framework que contém bibliotecas úteis à realização dos testes, de onde se destacam o JUnit, AssertJ, Hamcrest, Mockito e JsonPath (Spring.io, 2019b) por terem sido as mais utilizadas.

Nas secções seguintes são apresentados mais detalhes sobre os testes unitários e os testes de integração realizados.

## 7.1 Testes unitários

Os testes unitários têm como objetivo testar uma unidade do sistema individualmente. Quando uma unidade depende de outra é necessário isolar o comportamento dessa unidade e, para isso, usam-se *mocks* para substituir e simular a ação dos objetos que provocam dependências na unidade a ser testada. Em suma, um *mock* consiste em criar objetos para simular o comportamento de objetos reais. Assim, para simular os objetos pretendidos foi utilizada a anotação "@MockBean". Para a realização dos testes é também necessário criar alguns dados e instanciar objetos para serem usados, pelo que estes foram criados num método com a anotação "@Before", o que significa que estes dados são gerados antes da execução dos métodos de teste.

Nas próximas secções são apresentados alguns testes unitários realizados para cada camada do sistema.

### 7.1.1 Controladores

As classes designadas como controladores (*controllers*) recebem e emitem comunicações com o exterior e fazem a ligação dessas comunicações com os serviços da aplicação, pelo que é necessário criar *mocks* dos serviços existentes. Para simular os pedidos efetuados pelo exterior que espoletam a ação dos controladores foi usado o "MockMvc". O "MockMvc" permite simular o processamento real de um pedido HTTP sem ser necessário inicializar um servidor (Spring.io, 2019a). No Extrato de código 19 são apresentados os *mocks* criados para simular o comportamento dos serviços necessários para testar a classe "ContentController", usando a anotação "@MockBean" referida anteriormente.

```
@SpringBootTest
@AutoConfigureMockMvc
public class ContentControllerTest {

 @Autowired
 private MockMvc mvc;
 @MockBean
 private IContentService contentService;
 @MockBean
 private IUserService userService;
 @MockBean
 private ISolrService solrService;
 @MockBean
 private IHighlightedService highlightedService;
 @MockBean
 private IFavoriteService favoriteService;
 @MockBean
 private IHistoricService historicService;
 @MockBean
 private IReportService reportService;
 @MockBean
 private IPlaylistService playlistService;
 @MockBean
 private IRateService rateService;
 @MockBean
 private ITypeService typeService;
```

Extrato de código 19 - Declaração de *mocks* para os testes unitários da classe "ContentController"

No Extrato de código 20 é mostrado um dos testes unitários aplicados a um método do "ContentController", onde é efetuado um pedido para obter um conteúdo através do seu id.

```
@Test
public void getContent() throws Exception {
Mockito.when(contentService.getContent(1)).thenReturn(content1);

mvc.perform(get("/content/getContent?content_id=1")
.header("authorization", accessTokenUser1)
.contentType(MediaType.APPLICATION_JSON))
.andExpect(status().isOk())
.andExpect(jsonPath("$.title", is(content1.getTitle())));
}
```

Extrato de código 20 - Teste unitário para obter um conteúdo pelo seu id

Como se pode verificar no Extrato de código 20, é usada a biblioteca Mockito com o *mock* do "contentService" para simular o comportamento deste objeto, indicando que quando for chamado o método "getContent" deve ser retornado o conteúdo indicado. Depois, é usado o "MockMvc" para simular o pedido HTTP através do método "perform" e são verificados os resultados obtidos que, neste caso, consistem na verificação do código HTTP 200, correspondente ao *status* OK, e se o título do conteúdo obtido é o título esperado, usando para isso a biblioteca JsonPath. Para além disso, é também testada a segurança de acesso aos dados, uma vez que o "MockMvc" permite passar no pedido o cabeçalho de autorização com o respetivo *token* de acesso.

### 7.1.2 Serviços

A camada de serviços comunica com as classes modelo e com os repositórios, pelo que é necessário criar *mocks* dos repositórios para simular o seu comportamento real. No Extrato de código 21 são declarados *mocks* para simular o comportamento dos repositórios relativos à gestão de *Playlists* e à sua partilha.

```
@SpringBootTest
public class PlaylistServiceTest {

 @Autowired
 private IPlaylistService playlistService;
 @MockBean
 IPlaylistRepository playlistRepo;
 @MockBean
 IPlaylistShareRepository playlistShareRepo;
```

Extrato de código 21 - Declaração de *mocks* para os testes unitários da classe "PlaylistService"

No Extrato de código 22 é apresentado um teste da classe "PlaylistService" que tem como objetivo testar a adição de vários conteúdos a uma *Playlist* já existente no sistema.

```
@Test
 public void addMultipleContentsToPlaylist() throws Exception {
        List<Content> contentList = new ArrayList<>();
        contentList.add(content1);
        playlist4.setContents(contentList);

 Mockito.when(playlistRepo.findById(playlist4.getId())).thenReturn(Optional
.of(playlist4));
        Mockito.when(playlistRepo.save(playlist4)).thenReturn(playlist4);

        Playlist result =
playlistService.addMultipleContentsToPlaylist(playlist4.getId(), contents);

        assertEquals(playlist4, result);
        verify(playlistRepo, times(1)).findById(playlist4.getId());
        verify(playlistRepo, times(1)).save(playlist4);
}
```

Extrato de código 22 - Teste unitário para adicionar conteúdos a uma *Playlist*

O fluxo de execução do método real testado consiste em chamar um método do repositório de *Playlists* para obter a *Playlist* da base de dados através do seu id, atualizar os conteúdos dessa *Playlist,* adicionando-lhe os conteúdos pretendidos aos já existentes na sua lista de conteúdos e, por fim, atualizar a *Playlist* com os novos conteúdos na base de dados.

Assim, acompanhando o teste unitário realizado no Extrato de código 22, primeiramente é criada uma lista de conteúdos e é adicionado um conteúdo a essa lista. De seguida, a *Playlist* usada para realizar este teste ("playlist4") é atualizada com a lista de conteúdos criada previamente para mais tarde comparar com o resultado obtido. De forma a isolar o componente testado é usado o Mockito com o *mock* do repositório de *Playlists*, simulando assim dois comportamentos: retornar a *Playlist* usada neste teste quando é feito o pedido de pesquisa por id ao repositório; retornar a *Playlist* usada neste teste quando é executado o pedido de atualização da *Playlist* ao repositório. Após estas duas simulações é então executado o método alvo deste teste e recebido o resultado. Este resultado é então comparado com o resultado esperado (a "playlist4") usando a biblioteca JUnit e é também verificado se o método testado chamou exatamente uma vez o método de pesquisa por id e o método de armazenamento do repositório.

### 7.1.3 Repositórios

Para testar os métodos existentes nos repositórios (denominada por camada "dao" na API desenvolvida) é usada a própria base de dados Oracle em funcionamento, testando assim as operações de base de dados. De forma a não afetar os registos existentes na base de dados é usada a anotação "@DataJpaTest" nas classes de testes de repositórios, o que faz com que após a execução de cada método de testes as transações efetuadas na base de dados sejam revertidas. No Extrato de código 23 são visíveis as anotações usadas para configurar a classe de testes referente ao repositório de conteúdos.

```
@DataJpaTest
@AutoConfigureTestDatabase(replace = Replace.NONE)
public class ContentRepositoryTest {

 @Autowired
 IContentRepository contentRepo;
 @Autowired
 IUserAppRepository userRepo;
```

Extrato de código 23 - Configuração de classe de testes do "ContentRepository"

No  é apresentado um teste efetuado a um método da classe correspondente ao repositório de conteúdos ("ContentRepository"), que tem como objetivo obter da base de dados os três conteúdos mais recentes do utilizador passado por parâmetro.

```
@Test
```

```
 public void findTop3ByUserCreatedOrderByUploadDateDesc() throws Exception
{
        userRepo.save(user1);
        userRepo.save(user2);
        contentRepo.saveAll(contents);
        List<Content> expectedResult = new ArrayList<>();
        expectedResult.add(content6);
        expectedResult.add(content5);
        expectedResult.add(content1);

        List<Content> listResult =
contentRepo.findTop3ByUserCreatedOrderByUploadDateDesc(user1);

        assertEquals(expectedResult.size(), listResult.size());
        assertEquals(expectedResult.get(0).getTitle(),
listResult.get(0).getTitle());
        assertEquals(expectedResult.get(1).getTitle(),
listResult.get(1).getTitle());
        assertEquals(expectedResult.get(2).getTitle(),
listResult.get(2).getTitle());
}
```

Extrato de código 24 - Teste para obter os três conteúdos mais recentes do utilizador

Visualizando o extrato de código anterior do método testado, inicialmente são armazenados na base de dados utilizadores e conteúdos pertencentes a uma lista. Esses utilizadores e essa lista de conteúdos são criados antes da execução dos métodos, como referido anteriormente na secção 7. Depois de ter estes elementos armazenados na base de dados para serem usados neste teste, é criada uma lista de conteúdos e são-lhe adicionados os três conteúdos que representam o resultado esperado. Finalizada a criação de dados necessária para a execução do teste, é então invocado o método do repositório de conteúdos que espoleta uma *query* à base de dados para obter uma lista com os três conteúdos do utilizador passado por parâmetro, ordenados pela ordem de submissão mais recente. Com a lista de conteúdos obtida como resultado do método testado, são então feitas comparações entre o resultado esperado e o resultado recebido, verificando se o tamanho da lista obtida é igual ao tamanho da lista esperada, se os três conteúdos obtidos correspondem aos três conteúdos esperados e se estes estão pela ordem esperada na lista.

## 7.2 Testes de integração

Os testes de integração, ao contrário dos testes unitários em que os componentes são testados de forma individual, têm como objetivo testar o funcionamento dos componentes do sistema de forma integrada. No Extrato de código 25 são apresentadas as configurações usadas para o teste de integração da classe “ContentController”.

```
@SpringBootTest
@AutoConfigureMockMvc
@Transactional
public class ContentControllerIntegrationTest {

 @Autowired
```

```
private MockMvc mvc;
@Autowired
private IContentRepository contentRepo;
@Autowired
private IUserAppRepository userRepo;
@Autowired
private IHighlightedRepository highlightRepo;
@Autowired
private ITypeRepository typeRepo;
```

Extrato de código 25 – Configuração da classe para os testes de integração

A anotação "@Transactional" garante que as alterações efetuadas à base de dados são revertidas no fim de cada teste, não influenciando os dados existentes. Esta anotação e a inexistência de *mocks* são as principais diferenças em comparação com o teste unitário efetuado a este controlador (Extrato de código 19). Os testes de integração têm como alvo os controladores pois é aqui que são recebidos os pedidos HTTP vindos do exterior, sendo, portanto, este o ponto de partida de todas as ações executadas pelo sistema.

No  é apresentado um teste de integração efetuado ao "ContentController" para obter um conteúdo a partir do id recebido.

```
@Test
 public void getContent() throws Exception {
        Content contentTest = new
Content("title1","summary1","desc1","file1","thumbnail1","key1","author1","
contributor1","id1","type1","category1",
                      "format1","language1",new Date(), new Date(),
userRepo.findByUsernameIgnoreCase("david.trigueira"));
        contentTest.setContentStatistics(new ContentStatistics(0, 3, 0, 0,
contentTest));
        contentTest = contentRepo.save(contentTest);


 mvc.perform(get("/content/getContent?content_id="+contentTest.getId()+"")
                      .header("authorization", accessTokenUser1)
                      .contentType(MediaType.APPLICATION_JSON))
        .andExpect(status().isOk())
        .andExpect(jsonPath("$.id", is((int) contentTest.getId())));
}
```

Extrato de código 26 - Teste de integração para obter um conteúdo pelo seu id

De forma a ter dados para a execução do pedido é armazenado na base de dados um conteúdo criado para este teste. Depois do conteúdo estar armazenado, é então usado o "MockMvc" para simular o pedido HTTP à API para obter o conteúdo cujo id corresponde ao id passado por parâmetro nesse pedido. O pedido testado é o mesmo mencionado no teste unitário do Extrato de código 20, mas como se trata de um teste de integração não são usados *mocks* para simular comportamentos, integrando assim todos os componentes do sistema necessários para a concretização deste pedido, desde o seu ponto de entrada na API até à base de dados. Após a execução de todo o código, é verificado se a resposta ao pedido retorna o código HTTP 200 e se o id do conteúdo obtido corresponde ao id do conteúdo pedido.

# 8 Experimentação e Avaliação

Neste capítulo é apresentado o processo adotado para efetuar a experimentação e a avaliação do projeto desenvolvido e os respetivos resultados, limitações e conclusões desse processo.

Dado que o produto a desenvolver é algo com que o utilizador interage diretamente através da sua interface gráfica num navegador de internet, foi efetuada uma avaliação do grau de usabilidade do sistema desenvolvido. A usabilidade traduz-se em medir até que ponto um sistema, produto ou serviço pode ser usado por utilizadores específicos para atingir as metas estabelecidas com eficácia, eficiência e satisfação num contexto específico de uso (ISO, 2018).

A avaliação de usabilidade foi efetuada através de um questionário dirigido aos membros da equipa de *training* da ALERT, uma vez que estes são os elementos correspondentes à área de negócio do *software* desenvolvido e alguns participaram com sugestões no processo de levantamento de requisitos e funcionalidades a incluir no produto. Para além disso, estes elementos são também utilizadores da nova plataforma, tendo assim passado pela experiência de utilização das suas funcionalidades para responder ao questionário.

Nas próximas secções são apresentados mais detalhes sobre o questionário desenvolvido e os processos seguidos para efetuar a medição e avaliação da usabilidade.

## 8.1 Metodologia

Para medir a usabilidade da plataforma recorreu-se ao *System Usability Scale* (SUS) (Brooke, 1996). O SUS foi criado por John Brooke em 1986 e é um dos questionários de usabilidade mais usados, fornecendo uma visão global de avaliações de usabilidade. É constituído por dez itens com cinco opções de resposta do estilo Escala de Likert, permitindo conhecer o grau de conformidade do inquirido relativamente às afirmações propostas. Optou-se pelo SUS por diversos motivos (Martins, Rosa, Queirós, Silva, & Rocha, 2015):

- Disponibiliza uma pontuação única numa escala que é facilmente compreendida pela maior parte das pessoas;
- É confiável e robusto;
- É válido, isto é, é adequado para medir a usabilidade e fazer a distinção entre sistemas inutilizáveis e utilizáveis;
- É um questionário rápido e fácil de preencher;
- Abrange não só a usabilidade global do sistema (8 itens), mas também a sua facilidade de aprendizagem (2 itens);
- É uma ferramenta gratuita e rápida de aplicar.

Dado que as dez afirmações usadas no SUS se encontram escritas em Inglês, procedeu-se à alteração dessas afirmações tendo como base um estudo efetuado para a tradução e adaptação cultural deste questionário para a versão Portuguesa. Essa tradução é apresentada na Tabela 26.

Tabela 26 – Item original vs item correspondente em Português Europeu(Martins, Rosa, Queirós, Silva, & Rocha, 2015)

| Item Original | Item correspondente em Português Europeu |
|---|---|
| I think that I would like to use this sytem frequently. | Acho que gostaria de utilizar este produto com frequência. |
| I found the system unnecessarily complex. | Considerei o produto mais complexo do que necessário. |
| I thought the system was easy to use. | Achei o produto fácil de utilizar. |
| I think that I would need the support of a technical person to be able to use this system. | Acho que necessitaria de ajuda de um técnico para conseguir utilizar este produto. |
| I found the various functions in this system were well integrated. | Considerei que as várias funcionalidades deste produto estavam bem integradas. |
| I thought there was too much inconsistency in this system. | Achei que este produto tinha muitas inconsistências. |
| I would imagine that most people would learn to use this system very quickly. | Suponho que a maioria das pessoas aprenderia a utilizar rapidamente este produto. |
| I found the system very cumbersome to use. | Considerei o produto muito complicado de utilizar. |
| I felt very confident using the system. | Senti-me muito confiante a utilizar este produto. |

| Item Original | Item correspondente em Português Europeu |
|---|---|
| I needed to learn a lot of things before I could get going with this system. | Tive que aprender muito antes de conseguir lidar com este produto. |

Considerando esta tradução, substituiu-se a palavra "produto" por "plataforma" e utilizou-se então o questionário visível no Anexo B – Questionário de Usabilidade. Este questionário foi replicado no Google Forms para ser mais fácil a sua distribuição e recolha dos resultados.

Relativamente à métrica usada para medir a usabilidade da plataforma, esta consiste na pontuação final obtida após o uso de um processo específico de cálculos do SUS (explicado na secção 8.3) aplicado às respostas dadas pelos inquiridos. Através da pontuação calculada, é feito um enquadramento da usabilidade geral da plataforma tendo em conta a tabela de valores de referência de resultados do SUS (Tabela 28) que permite interpretar a pontuação final obtida.

Quanto às pessoas inquiridas para o questionário, dado o objetivo deste estudo, a população são os colaboradores da ALERT e a população alvo os colaboradores da ALERT, pertencentes à equipa de *training*, e que utilizam a plataforma desenvolvida. A amostragem é não probabilística (amostra de conveniência), tendo sido selecionados onze elementos com idades compreendidas entre os 22 e os 41 anos.

## 8.2 Execução

De forma a colocar em execução a realização do questionário houve, previamente, uma comunicação da existência de um questionário com o líder da equipa de *training* e foram reunidos os contactos de todos os utilizadores da plataforma aptos para responderem ao questionário. Posteriormente, foi enviada uma mensagem de correio eletrónico para onze pessoas com a hiperligação para o endereço do questionário online, pedindo a sua colaboração e indicando que o questionário estaria disponível durante três dias até ser fechado.

## 8.3 Resultados

Nesta secção é efetuada a análise das respostas dadas ao questionário, apresentando os resultados dos cálculos realizados para avaliar a usabilidade da plataforma e respetivas conclusões.

Terminada a data de realização do inquérito procedeu-se ao tratamento dos dados recolhidos. A taxa de participação foi de 100%, tendo todos os onze inquiridos respondido ao questionário e a todas as questões colocadas. Assim, para cada questão, foram efetuadas onze atribuições de pontuação. Na Tabela 27 são apresentados os dados recolhidos das respostas dadas ao

questionário mostrando, para cada questão, o número de votos de cada resposta, destacando a negrito a mais votada.

Tabela 27 - Dados das respostas do questionário

| | Questão | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| 1 | Acho que gostaria de utilizar esta plataforma com frequência. | 0 (0%) | 0 (0%) | 0 (0%) | 1 (9,1%) | **10 (90,9%)** |
| 2 | Considerei a plataforma mais complexa do que necessário. | **9 (81,8%)** | 2 (18,2%) | 0 (0%) | 0 (0%) | 0 (0%) |
| 3 | Achei a plataforma fácil de utilizar. | 0 (0%) | 0 (0%) | 0 (0%) | 1 (9,1%) | **10 (90,9%)** |
| 4 | Acho que necessitaria de ajuda de um técnico para conseguir utilizar esta plataforma. | **8 (72,7%)** | 3 (27,3%) | 0 (0%) | 0 (0%) | 0 (0%) |
| 5 | Considerei que as várias funcionalidades desta plataforma estavam bem integradas. | 0 (0%) | 0 (0%) | 0 (0%) | 4 (36,4%) | **7 (63,6%)** |
| 6 | Achei que esta plataforma tinha muitas inconsistências. | **9 (81,8%)** | 1 (9,1%) | 1 (9,1%) | 0 (0%) | 0 (0%) |
| 7 | Suponho que a maioria das pessoas aprenderia a utilizar rapidamente esta plataforma. | 0 (0%) | 0 (0%) | 0 (0%) | 3 (27,3%) | **8 (72,7%)** |
| 8 | Considerei a plataforma muito complicada de utilizar. | **9 (81,8%)** | 2 (18,2%) | 0 (0%) | 0 (0%) | 0 (0%) |
| 9 | Senti-me muito confiante a utilizar esta plataforma. | 0 (0%) | 0 (0%) | 1 (9,1%) | 2 (18,2%) | **8 (72,7%)** |
| 10 | Tive que aprender muito antes de conseguir lidar com esta plataforma. | **7 (63,6%)** | 2 (18,2%) | 2 (18,2%) | 0 (0%) | 0 (0%) |

Para obter a pontuação final do SUS foi seguido o processo referido pelo seu inventor, que consiste nos passos seguintes (Brooke, 1996), tendo em conta que a pontuação de cada questão varia entre 0 e 4:

- Somar as pontuações das questões ímpares, considerando que para cada questão ímpar a pontuação consiste em subtrair 1 ao valor da escala atribuído pelo inquirido;

- Somar as pontuações das questões pares, considerando que para cada questão par a pontuação consiste em subtrair ao valor 5 o valor da escala atribuído pelo inquirido;

- Somar as pontuações das questões ímpares e pares;

- Multiplicar a soma das pontuações por 2.5 para converter a pontuação original numa escala de 0-40 para 0-100, obtendo assim o valor global da usabilidade do sistema.

Depois de obtida a pontuação final e de forma a interpretá-la, esta deve ser comparada com a tabela de valores de referência de resultados do SUS, apresentada na Tabela 28.

Tabela 28 – Valores de referência para interpretação da pontuação SUS (Alathas, 2018)

| Pontuação SUS | Nota | Adjetivo de classificação |
|---|---|---|
| $> 80.3$ | A | Excelente |
| $68 – 80.3$ | B | Bom |
| $68$ | C | Aceitável |
| $51 – 68$ | D | Fraco |
| $< 51$ | F | Péssimo |

Note-se que, apesar do intervalo da pontuação ser de 0 a 100, os resultados não refletem percentagens e por isso devem ser interpretados com base nos valores de referência, sendo que o valor correspondente à classificação média (50º percentil) e considerada “Aceitável” na pontuação SUS é de 68.

A partir das respostas obtidas foi então aplicado o processo descrito acima com o auxílio do Excel, como forma de automatizar os cálculos necessários através do uso de fórmulas. Recorrendo a esta ferramenta foram inseridas as pontuações atribuídas por cada um dos onze participantes às questões, como se pode visualizar na Figura 69.

**SUS Scoring Sheet & Reliability Test**

*SUS data (scored 1=Strongly disagree; 5=Strongly agree)*

*Average score is 68*

| | Q1 | Q2 | Q3 | Q4 | Q5 | Q6 | Q7 | Q8 | Q9 | Q10 |
|---|---|---|---|---|---|---|---|---|---|---|
| Participants | I think that | I found the | I thought | I think | I found | I thought | I would | I found | I felt very | I needed |
| 1 | 4 | 1 | 5 | 2 | 4 | 3 | 4 | 2 | 4 | 2 |
| 2 | 5 | 2 | 4 | 1 | 4 | 2 | 4 | 1 | 5 | 2 |
| 3 | 5 | 1 | 5 | 1 | 5 | 1 | 5 | 1 | 5 | 1 |
| 4 | 5 | 1 | 5 | 1 | 5 | 1 | 5 | 1 | 5 | 1 |
| 5 | 5 | 1 | 5 | 1 | 5 | 1 | 5 | 1 | 5 | 1 |
| 6 | 5 | 1 | 5 | 1 | 5 | 1 | 5 | 1 | 5 | 1 |
| 7 | 5 | 2 | 5 | 2 | 5 | 1 | 4 | 2 | 4 | 3 |
| 8 | 5 | 1 | 5 | 1 | 5 | 1 | 5 | 1 | 5 | 1 |
| 9 | 5 | 1 | 5 | 1 | 4 | 1 | 5 | 1 | 5 | 1 |
| 10 | 5 | 1 | 5 | 1 | 5 | 1 | 5 | 1 | 5 | 1 |
| 11 | 5 | 1 | 5 | 2 | 4 | 1 | 5 | 1 | 3 | 3 |

Figura 69 – Inserção das respostas dos inquiridos no Excel

Depois de inseridas as respostas dadas, é calculada a pontuação SUS para cada participante seguindo o processo descrito anteriormente e é calculada a média final da pontuação tendo em conta as pontuações de todos os inquiridos. Na Figura 70 são apresentados os resultados obtidos.

| Scales | | | |
|---|---|---|---|
| Odd items | Even items | SUS score (/100) | Grades |
| 16 | 15 | 77,5 | B |
| 17 | 17 | 85 | A |
| 20 | 20 | 100 | A |
| 20 | 20 | 100 | A |
| 20 | 20 | 100 | A |
| 20 | 20 | 100 | A |
| 18 | 15 | 82,5 | A |
| 20 | 20 | 100 | A |
| 19 | 20 | 97,5 | A |
| 20 | 20 | 100 | A |
| 17 | 17 | 85 | A |
| | **Average score** | 93,40909091 | A |

Figura 70 – Resultados da pontuação SUS

Analisando os resultados de cada inquirido, verifica-se que todos estão satisfeitos com a usabilidade da plataforma uma vez que nenhuma das pontuações é inferior ao valor aceitável de usabilidade (68), sendo a menor pontuação de 77,5, o que corresponde a uma classificação "Bom". Todas as restantes pontuações obtidas inserem-se na classificação "Excelente".

Relativamente à pontuação final geral do SUS, obteve-se uma pontuação média de aproximadamente 93.41.

## 8.4 Limitações

Na realização deste estudo foram encontradas algumas limitações, nomeadamente no tipo de amostragem. Ao adotar a técnica de amostragem por conveniência está-se a limitar a capacidade de fazer afirmações gerais com rigor estatístico sobre a população e a precisão dos resultados.

Outra limitação reside no facto de ao aplicar este questionário não ser possível obter informações diagnósticas, isto é, o SUS possibilita medir a usabilidade e compará-la com valores de referência, mas nenhum dos itens do questionário permite determinar o que deve ser melhorado em específico na plataforma, obtendo-se apenas uma avaliação geral.

## 8.5 Conclusão

Com o estudo efetuado obteve-se uma pontuação final de aproximadamente 93.41, o que permite concluir que, enquadrando o resultado obtido com os valores de referência apresentados na Tabela 28, a usabilidade da plataforma pode ser classificada como “Excelente” com nota A.

Assim, pode-se concluir que a plataforma desenvolvida apresenta alta usabilidade e que os inquiridos ficaram muito satisfeitos com o seu uso e voltariam a usá-la novamente, o que se confirma visto que a plataforma já é usada atualmente na organização.

# 9 Conclusão

Neste capítulo é efetuada uma reflexão sobre os objetivos cumpridos na concretização desta dissertação e do projeto, as limitações encontradas e o trabalho futuro, terminando com uma apreciação final sobre o trabalho desenvolvido.

## 9.1 Objetivos concluídos

Relativamente aos objetivos propostos, o objetivo principal que consistia em desenvolver uma nova plataforma *web,* com uma forte componente colaborativa para a gestão de diferentes tipos de conteúdos de aprendizagem, foi alcançado com sucesso. A investigação sobre o estado da arte de sistemas de Bibliotecas Multimédia existentes e de tecnologia pertinente para o desenvolvimento do projeto foi muito útil, uma vez que se seguiram algumas das conclusões retiradas desse estudo para a construção da solução. Os requisitos inicialmente definidos foram cumpridos e estão em atual funcionamento no novo sistema, existindo ainda margem de melhoria em alguns deles. Uma das principais preocupações deste projeto resumia-se à usabilidade da plataforma. Tendo em consideração o seu estado no momento da experimentação e avaliação efetuada, verificou-se que a usabilidade da plataforma pode ser classificada como "Excelente", pelo que se pode afirmar que esse objetivo também foi cumprido.

## 9.2 Limitações e trabalho futuro

No desenvolvimento deste projeto verificaram-se algumas limitações. Uma vez que a plataforma desenvolvida é um produto em progressão e desenvolvimento contínuo, naturalmente vão surgindo novas ideias e possíveis requisitos durante a sua implementação,

pelo que foi identificado trabalho futuro a realizar de forma a introduzir novas funcionalidades relevantes e a complementar as já existentes.

Relativamente à compatibilidade da plataforma, verificaram-se alguns obstáculos no que diz respeito ao uso do *Internet Explorer*, uma vez que também era requerida compatibilidade com este navegador. Dado o facto de este ser um navegador descontinuado, tecnologicamente inferior e com menor suporte do que o outro principal navegador utilizado no desenvolvimento (*Google Chrome),* houve um trabalho adicional em solucionar problemas que surgiram no *Internet Explorer* sem surgirem no *Google Chrome*.

A nível das funcionalidades implementadas, regista-se uma limitação no conceito desenhado quanto aos tipos e categorias de um conteúdo que deve ser modificado. Inicialmente, projetou-se que um conteúdo tem obrigatoriamente um tipo e pode ter uma categoria, mas na fase final da implementação detetou-se que deve existir a possibilidade de um conteúdo ser associado a vários tipos e a várias categorias. Outro trabalho futuro a realizar e relevante consiste na criação de uma área na plataforma onde seja possível um administrador gerir os tipos e categorias de conteúdos existentes pois, atualmente, a gestão dos tipos e categorias tem de ser efetuada diretamente na base de dados por alguém com conhecimento técnico para tal.

Na realização da Experimentação e Avaliação foram encontradas algumas limitações, referidas na secção 8.4, que consistem no facto da técnica de amostragem adotada ter sido uma técnica de amostragem por conveniência e no facto do uso do SUS não permitir obter informações diagnósticas sobre o que deve ser melhorado em específico na plataforma.

## 9.3 Apreciação final

Concluindo, o balanço final do trabalho desenvolvido é muito positivo visto que contribuiu com sucesso com uma solução para o problema proposto e, consequentemente, resultou numa proposta de emprego para integrar a organização e continuar o trabalho elaborado. A nova plataforma *web* multimédia está atualmente disponível para os colaboradores da ALERT e permite então que sejam os próprios utilizadores a publicarem e efetuarem a gestão de conteúdos na plataforma. Assim, é disponibilizado um meio de visualização e partilha de recursos através de uma plataforma moderna e atual com alta usabilidade, facilitando os processos de ensino e aprendizagem.

# 10 Referências


Al Owsi, B. (2010). E-learning content. *Journal Of AL-Turath University College*(8), 189-198. Obtido de https://www.iasj.net/iasj?func=article&aId=36729

Alathas, H. (19 de Novembro de 2018). *How to Measure Product Usability with the System Usability Scale (SUS) Score*. Obtido em Junho de 2019, de UX Planet: https://uxplanet.org/how-to-measure-product-usability-with-the-system-usability-scale-sus-score-69f3875b858f

ALERT. (28 de 11 de 2018). *ALERT Life Sciences Computing*. Obtido de ALERT® ONLINE: http://www.alert-online.com/pt/mission-values

Annesley, C. (27 de Outubro de 2017). *Airbus saves millions after building a digital learning library*. Obtido em Novembro de 2018, de ComputerWeekly.com: https://www.computerweekly.com/news/450428969/Airbus-saves-millions-after-building-a-digital-learning-library

Apache. (30 de Novembro de 2018). *Solr Features*. Obtido de Apache Solr: http://lucene.apache.org/solr/features.html

Apache Software Foundation. (2018). *Apache Subversion*. Obtido em Janeiro de 2019, de Subversion.apache.org: https://subversion.apache.org/

Atlassian. (2019). *Jira | Issue & Project Tracking Software | Atlassian*. Obtido em Janeiro de 2019, de Atlassian: https://www.atlassian.com/software/jira

Bankier, J. G., & Gleason, K. (2014). *Institutional Repository Software Comparison.* Paris, França: UNESCO. Obtido em Novembro de 2018, de https://unesdoc.unesco.org/ark:/48223/pf0000227115

Becker, P.-N. (08 de Novembro de 2018). *Home - DSpace - DuraSpace Wiki*. Obtido em Dezembro de 2018, de Wiki.duraspace.org: https://wiki.duraspace.org/display/DSPACE/

Biblioteca Mundial. (2018a). *Navegar por Período - Biblioteca Digital Mundial*. Obtido em Dezembro de 2018, de Wdl.org: https://www.wdl.org/pt/time/

Biblioteca Mundial. (2018b). *Página Inicial da Biblioteca Digital Mundial*. Obtido em Dezembro de 2018, de Wdl.org: https://www.wdl.org/pt/

Biblioteca Mundial. (2018c). *Resultados da Pesquisa - Biblioteca Digital Mundial*. Obtido em Dezembro de 2018, de Wdl.org: https://www.wdl.org/pt/search/?q=Portugal&regions=europe&qla=pt

Brooke, J. (1996). SUS - A quick and dirty usability scale. Em P. Jordan, B. Thomas, B. Weerdmeester, & I. McClelland, *Usability Evaluation in Industry* (pp. 189-194). London: Taylor and Francis.

Comissão Europeia. (18 de Novembro de 2010). *European Comission - Europeana - Europe's digital library: frequently asked questions*. Obtido em Dezembro de 2018, de European Comission: http://europa.eu/rapid/press-release_MEMO-10-586_en.htm?locale=en

Correia, J. (2016). *Indexação de Documentos Clínicos.*

DB-Engines. (Dezembro de 2018a). *DB-Engines Ranking*. Obtido em Dezembro de 2018, de DB-Engines: https://db-engines.com/en/ranking/search+engine

DB-Engines. (Dezembro de 2018b). *DB-Engines Ranking - Method*. Obtido em Dezembro de 2018, de DB-Engines: https://db-engines.com/en/ranking_definition

DB-Engines. (Dezembro de 2018c). *Elasticsearch vs. Solr Comparison*. Obtido em Dezembro de 2018, de DB-Engines: https://db-engines.com/en/system/Elasticsearch%3BSolr

DB-Engines. (Dezembro de 2018d). *Historical trend of search engines popularity*. Obtido em Dezembro de 2018, de DB-Engines: https://db-engines.com/en/ranking_trend/search+engine

*DEI - Tese/Dissertação/Estágio*. (2018). Obtido de Moodle ISEP: https://moodle.isep.ipp.pt/course/view.php?id=6829

Don, K. J., Bainbridge, D., & Witten, I. H. (2002). *The design of Greenstone 3: An agent based dynamic digital library.* Technical report, University of Waikato, Department of Computer Science, Hamilton New Zealand. Obtido em Dezembro de 2018, de http://www.greenstone.org/manuals/gs3design.pdf

Donohue, T. (17 de Março de 2015a). *Application Layer - DSpace 6.x Documentation - DuraSpace Wiki*. Obtido em Dezembro de 2018, de Wiki.duraspace.org:

https://wiki.duraspace.org/display/DSDOC6x/Application+Layer#ApplicationLayer-ServletsandJSPs(JSPUIOnly)

Donohue, T. (17 de Março de 2015b). *Architecture - DSpace 6.x Documentation - DuraSpace Wiki*. Obtido em Dezembro de 2018, de Wiki.duraspace.org: https://wiki.duraspace.org/display/DSDOC6x/Architecture

Donohue, T. (22 de Fevereiro de 2016c). *Business Logic Layer - DSpace 6.x Documentation - DuraSpace Wiki*. Obtido em Dezembro de 2018, de Wiki.duraspace.org: https://wiki.duraspace.org/display/DSDOC6x/Business+Logic+Layer#BusinessLogicLayer-ContentManagementAPI

Donohue, T. (15 de Agosto de 2017d). *Importing and Exporting Items via Simple Archive Format - DSpace 5.x Documentation - DuraSpace Wiki*. Obtido em Dezembro de 2018, de Wiki.duraspace.org: https://wiki.duraspace.org/display/DSDOC5x/Importing+and+Exporting+Items+via+Simple+Archive+Format

Donohue, T., & Pottinger, H. (14 de Dezembro de 2017). *XMLUI Configuration and Customization - DSpace 6.x Documentation - DuraSpace Wiki*. Obtido em Novembro de 2018, de Wiki.duraspace.org: https://wiki.duraspace.org/display/DSDOC6x/XMLUI+Configuration+and+Customization

DSpace. (2018). *Technical Specifications - Fedora*. Obtido em Dezembro de 2018, de duraspace: https://duraspace.org/fedora/resources/technical-specifications/

DuraSpace. (2018). *Technical Specifications - DSpace*. Obtido em Dezembro de 2018, de duraspace: https://duraspace.org/dspace/resources/technical-specifications/

DuraSpace. (2019). *About Fedora - Fedora*. Obtido em Fevereiro de 2019, de Duraspace.org: https://duraspace.org/fedora/about/

Eeles, P. (15 de Novembro de 2005). *Capturing Architectural Requirements*. Obtido em Janeiro de 2019, de Ibm.com: https://www.ibm.com/developerworks/rational/library/4706.html#N100A7

Europeana. (2018a). *Europeana 1914-1918 - Europeana Collections*. Obtido em Dezembro de 2018, de Europeana Collections: https://www.europeana.eu/portal/pt/collections/world-war-I/contribute#action=users/account

Europeana. (2018b). *Europeana 1914-1918 - Europeana Collections*. Obtido em Dezembro de 2018, de Europeana Collections: https://www.europeana.eu/portal/pt/collections/world-war-I/contribute#action=contributions/21810/attachments/new

Europeana. (2018c). *Europeana 1914-1918 - Europeana Collections*. Obtido em Dezembro de 2018, de Europeana Collections: https://www.europeana.eu/portal/pt/collections/world-war-I/contribute#action=contributor

Europeana. (2018d). *Europeana Collections*. Obtido em Dezembro de 2018, de Europeana Collections: https://www.europeana.eu/portal/pt

Europeana. (2018e). *Pessoas - Europeana Collections*. Obtido em Dezembro de 2018, de Europeana Collections: https://www.europeana.eu/portal/pt/explore/people.html?theme=art

Europeana. (2018f). *Procurar resultados - Europeana Collections*. Obtido em Dezembro de 2018, de Europeana Collections: https://www.europeana.eu/portal/pt/collections/world-war-I?q=&view=grid

Fielding, R. T. (2000). Chapter 5: Representational State Transfer (REST). Em R. T. Fielding, *Architectural styles and the design of network-based software architectures* (pp. 76-105). Irvine: University of California. Obtido em Fevereiro de 2019, de https://www.ics.uci.edu/~fielding/pubs/dissertation/fielding_dissertation.pdf

Fuchs, M., Muscogiuri, C., Hemmje, M., & Niederée, C. (Agosto de 2004). Digital libraries in knowledge management: an e-learning case study. *International Journal on Digital Libraries*(1), 31-35. Obtido em Novembro de 2018, de https://link.springer.com/article/10.1007/s00799-003-0060-x

Gospodnetic, O., Hatcher, E., & McCandless, M. (2010). *Lucene in Action.* Manning Publications.

Grainger, T., & Potter, T. (2014). *Solr in Action.* Manning Publications.

Greenstone. (2015a). *Factsheet :: Greenstone Digital Library Software*. Obtido em Janeiro de 2019, de Greenstone.org: http://www.greenstone.org/factsheet

Greenstone. (2015b). *Welcome :: Greenstone Digital Library Software*. Obtido em Janeiro de 2019, de Greenstone.org: http://www.greenstone.org/

Greenstone. (Novembro de 2018). *index - Greenstone*. Obtido em Dezembro de 2018, de Wiki.greenstone.org: http://wiki.greenstone.org/doku.php?id=index

grokonez. (17 de Outubro de 2018). *Spring Security - JWT Authentication Architecture | Spring Boot*. Obtido em Junho de 2019, de grokonez: https://grokonez.com/spring-framework/spring-security/spring-boot-spring-security-jwt-authentication-architecture-tutorial#Retrieve_User_details_with_UserDetailsService

Hinman, M. L., Gheorghe, R., & Russo, R. (2015). *Elasticsearch in Action.* Manning Publications.

International Business Machines. (2001). *Rational Unified Process. Best Practices for Software Development Teams.* Rational Software White Paper.

International Business Machines. (5 de Fevereiro de 2019). *Java Persistence API (JPA)*. Obtido em Fevereiro de 2019, de Ibm.com: https://www.ibm.com/support/knowledgecenter/en/SSAW57_liberty/com.ibm.websphere.wlp.nd.multiplatform.doc/ae/cwlp_jpa.html

ISO. (2018). *Ergonomics of human-system interaction — Part 11: Usability: Definitions and concepts*. Obtido em Fevereiro de 2019, de Iso.org: https://www.iso.org/obp/ui/#iso:std:iso:9241:-11:ed-2:v1:en

Koen, P., Ajamian, G., Boyce, S., Clamen, A., Fisher, E., Fountoulakis, S., . . . Seibert, R. (2002). *Fuzzy Front End: Effective Methods, Tools, and Techniques.*

Kökörčený, M. (2011). Architecture and flexibility of digital libraries systems. *International Journal of Systems Applications, Engineering & Development, 5*(2), 237-244.

Kuc, R. (Junho de 2017). *Solr vs. Elasticsearch*. Obtido de DZone: https://dzone.com/articles/top-15-solr-vs-elasticsearch-differences-1

Leonard, A. (2013). *Academic libraries in support of e-learning: The case of University of Namibia Library.* Obtido de https://www.researchgate.net/publication/318130083_Academic_libraries_in_support_of_e-learning_The_case_of_University_of_Namibia_Library

Mark, J., & Woods, A. (10 de Dezembro de 2018). *Fedora Repository Home - Fedora Repository - DuraSpace Wiki*. Obtido em Dezembro de 2018, de Wiki.duraspace.org: https://wiki.duraspace.org/display/FF

Martins, A. I., Rosa, A. F., Queirós, A., Silva, A., & Rocha, N. (2015). European portuguese validation of the system usability scale (SUS). *Procedia Computer Science 67* (pp. 293-300). Elsevier B.V. doi:10.1016/j.procs.2015.09.273.

Morajkar, A. (30 de Novembro de 2018). *Building Search Interface Using Apache Solr*. Obtido de Devx: http://www.devx.com/opensource/building-search-interface-using-apache-solr-in-dotnet.html

Nicola, S. (2018). Análise_Valor_Aula_4_20NOV_2018_1hora_AHP.

Pyrounakis, G., & Nikolaidou, M. (2009). *Comparing Open Source Digital Library Software.*

Rich, N., & Holweg, M. (2000). *Value Analysis, Value Engineering.*

Rupesh, A. K. (23 de Novembro de 2017). Digital Library Software. Tumkur, Karnataka, India. Obtido em Novembro de 2018, de https://pt.slideshare.net/tudlis/digital-library-software

Saaty, T. L. (2008). Decision making with the Analytic Hierarchy Process. *International Journal of Services Sciences, 1*(1), 83-98.

Sastry, H. G., & Reddy, L. C. (2010). Significance of Web 2.0 in Digital Libraries. *International Journal on Computer Science and Engineering, 2*(6), 2208-2211. Obtido de https://pdfs.semanticscholar.org/3059/17d2421a30a9decbd7f782c9ff026b9baeb1.pdf

Smith, M., Barton, M., Bass, M., Branschofsky, M., McClellan, G., Stuve, D., . . . Walker, J. H. (2003). DSpace - An Open Source Dynamic Digital Repository. *D-Lib Magazine, 9*(1). Obtido de http://dlib.org/dlib/january03/smith/01smith.html

Spring.io. (2019a). *Testing the Web Layer*. Obtido em Março de 2019, de Spring.io: https://spring.io/guides/gs/testing-web/

Spring.io. (2019b). *46. Testing.* Obtido em Março de 2019, de Docs.spring.io: https://docs.spring.io/spring-boot/docs/current/reference/html/boot-features-testing.html

Staples, T., Wayland, R., & Payette, S. (2003). The Fedora Project - An Open-Source Digital Object Repository Management System. *D-Lib Magazine, 9*(4). Obtido de http://mirror.dlib.org/dlib/april03/staples/04staples.html

Tan, K. (Maio de 2018). *Apache Solr vs Elasticsearch - the Feature Smackdown!* Obtido em Dezembro de 2018, de Solr-vs-elasticsearch.com: http://solr-vs-elasticsearch.com/

The Fedora Development Team. (23 de Julho de 2008). *Fedora Tutorial 1 - Introduction to Fedora*. Obtido em Dezembro de 2018, de wiki.duraspace: https://wiki.duraspace.org/display/FEDORACREATE/Tutorial+1+-+Introduction+to+Fedora#Tutorial1-IntroductiontoFedora-Section5:TheContentModelArchitecture

TortoiseSVN. (2018). *Home · TortoiseSVN*. Obtido em Janeiro de 2019, de Tortoisesvn.net: https://tortoisesvn.net/

Tramboo, S., Humma, & Shafi, S. (2012). A Study on the Open Source Digital Library Software's: Special Reference to DSpace, EPrints and Greenstone. *International Journal Of Computer Applications, 59*(16).

Verma, L., & Kumar, N. (Setembro de 2018). Comparative Analysis of Open Source Digital Library Softwares: A Case Study. *DESIDOC Journal of Library & Information Technology, 38*(5), 361-368. doi:https://doi.org/10.14429/djlit.38.5.12425

Warwick Manufacturing Group. (2007). *Product Excellence using Six Sigma: Quality Function Deployment.*

Woodall, T. (2003). Conceptualising 'Value for the Customer': An Attributional, Structural and Dispositional Analysis. *Academy of marketing science review, 12*, 1-42.

World Digital Library. (2018). *About the World Digital Library*. Obtido em Dezembro de 2018, de World Digital Library: https://www.wdl.org/en/about/

Zagalo, H. T., Martins, J. A., Almeida, P. M., & Pinto, J. S. (2003). *A Contribution of Open Source Technologies to Support Distributed Digital Library's Repository and Index Services.* Instituto de Engenharia Electrónica e Telemática de Aveiro. Obtido de https://www.researchgate.net/publication/266471607_A_Contribution_of_Open_Source_Technologies_to_Support_Distributed_Digital_Library's_Repository_and_Index_Services

Zeithaml, V. A. (1988). Consumer Perceptions of Price, Quality, and Value: A Means-End Model and Synthesis of Evidence. *Journal of Marketing, 52*, 2-22. doi:https://doi.org/10.2307/1251446

# 11 Anexo A – Indicadores de Benefícios e Sacríficios

| BENEFITS | | SACRIFICES |
|---|---|---|
| **Attributes** | **Outcomes** | |
| Perceived quality | Functional benefits | Price |
| Product quality | Utility | Market price |
| Quality | Use function | Monetary costs |
| Service quality | Aesthetic function | Financial |
| Technical quality | Operational benefits | Costs |
| Functional quality | Economy | Costs of use |
| Performance quality | Logistical benefits | Perceived costs |
| Service performance | Product benefits | Search costs |
| Service | Strategic benefits | Acquisition costs |
| Service support | Financial benefits | Opportunity costs |
| Special service aspects | Results for the customer | Delivery and installation costs |
| Additional services | Social benefits | Costs of repair |
| Core solution | Security | Training and maintenance costs |
| Customisation | Convenience | Non-monetary costs |
| Reliability | Enjoyment | Non-financial costs |
| Product characteristics | Appreciation from users | Relationship costs |
| Product attributes | Knowledge, humour | Psychological costs |
| Features | Self-expression | Time |
| Performance | Personal benefits | Human energy |
| | Association with social groups | Effort |
| | Affective arousal | |

Figura 71 – Indicadores de Benefícios e Sacríficos (Woodall, 2003)

# 12 Anexo B – Questionário de Usabilidade

No âmbito da Tese de Mestrado sobre o tema **Plataforma eLEARNING e Biblioteca Multimédia**, desenvolvida no ramo de Engenharia de Software do Departamento de Engenharia Informática (DEI) do Instituto Superior de Engenharia do Porto (ISEP), venho por este meio pedir a vossa colaboração no preenchimento deste questionário.

Este questionário consiste no questionário System Usability Scale (SUS) e tem como objetivo avaliar o grau de **usabilidade** da plataforma desenvolvida durante a Tese, a ALERT® eLEARNING Library.

O questionário é constituído por **10** afirmações e todas devem ser **obrigatoriamente respondidas** tendo em conta uma escala de **1** a **5**, sendo que **1** significa **discordo completamente** e **5 concordo completamente**. Por favor responda rapidamente, sem pensar demasiado sobre cada item.

As respostas são confidenciais e serão objeto de tratamento estatístico. Em caso de dúvida poderá contactar-me através do Skype ou do correio eletrónico.

Obrigado,

David Trigueira

1. Acho que gostaria de utilizar esta plataforma com frequência.

2. Considerei a plataforma mais complexa do que necessário.

3. Achei a plataforma fácil de utilizar.
4. Acho que necessitaria de ajuda de um técnico para conseguir utilizar esta plataforma.
5. Considerei que as várias funcionalidades desta plataforma estavam bem integradas.
6. Achei que esta plataforma tinha muitas inconsistências.
7. Suponho que a maioria das pessoas aprenderia a utilizar rapidamente esta plataforma.
8. Considerei a plataforma muito complicada de utilizar.
9. Senti-me muito confiante a utilizar esta plataforma.
10. Tive que aprender muito antes de conseguir lidar com esta plataforma.